UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Implantación de mejoras de usabilidad y funcionalidad en una plataforma de gestión de proyectos

Autor/es

GERMÁN RUBIO RUIDIAZ

Director/es

BEATRIZ PÉREZ VALLE

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Ingeniería Informática

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2017-18

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

Facultad de Ciencia y Tecnología

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Implantación de mejoras de usabilidad y funcionalidad en una plataforma de gestión de proyectos.

Realizado por:

Germán Rubio Ruidiaz

Tutelado por:

Beatriz Pérez Valle

Logroño, Junio, 2018

ÍNDICE

Resumen

La empresa Netbrain Solutions S.L. utiliza una aplicación web para gestionar y controlar el tiempo que invierte cada trabajador en los proyectos que se desarrollan en la empresa.

Este trabajo ha sido propuesto por Hernán González, director de proyectos de la empresa, con el fin de mejorar la aplicación que utilizan actualmente, tratándose pues de un trabajo de reingeniería.

El punto principal, y en el que se basa prácticamente todo el proyecto, es realizar mejoras de usabilidad de dicha aplicación, de manera que el uso diario de la misma sea más sencillo e intuitivo. También se han llevado a cabo diversas tareas de mejora de la funcionalidad de la aplicación ya existente. A parte de todo esto, se ha solucionado un conjunto de errores existentes que impiden el uso correcto de la aplicación.

Como resultado de la realización de este proyecto, se ha conseguido un producto que alcanza las expectativas del cliente y que, a su vez, utiliza una gran variedad de tecnologías.

Abstract

The company Netbrain Solutions Ltd. uses a web application to manage and control the time spent by each employee on the projects developed within the company.

This project has been proposed by Hernán González, who is the project director of the company, with the aim of improving the current application used by the company, therefore it is considered a reengineering project.

The main point, which the project is mainly based on, consists in making usability improvements so as to make the diary use of the application more user-friendly and easier. Besides, several improvements in the functionality of the current applitacion have been carried out. Apart from this, a set of problems, which prevented the application from working properly, has been solved.

As a result of the achievement of this project, we have obtained a product which fulfils the client expectations and, at the same time, uses a great variety of technologies.

1. Introducción

La siguiente memoria tiene como objetivo recoger los puntos más importantes del Trabajo Fin de Grado que lleva por título "Implantación de mejoras de usabilidad y funcionalidad en una plataforma de gestión de proyectos", desarrollado en la empresa *Netbrain Solutions S.L. .*

1.1 Contexto

La empresa *Netbrain Solutions S.L.* dispone de una herramienta, llamada *TimeTracker,* para el control del tiempo dedicado por los empleados en el desarrollo de los proyectos. Dicha herramienta se presenta como una aplicación web y se utiliza a diario tanto por los trabajadores como por los responsables de la empresa.

La aplicación está dividida en varias páginas, cada una de las cuales será considerada como un *módulo*. En particular, entre dichos módulos destacamos los siguientes por su interés en este trabajo:

- Módulo de estadísticas: donde aparecerán las gráficas y estadísticas de cada proyecto que se desarrolla en la empresa.
- Módulo de proyectos: donde aparecerán listados todos los proyectos de la empresa.
- Página principal: donde los trabajadores introducen diariamente las horas dedicadas a cada tarea.

Los roles con los que se puede acceder a la aplicación son dos:

- User: este tipo de usuario tiene la posibilidad de introducir las horas que ha trabajado en su jornada laboral. Además puede crear y modificar tanto proyectos web, como las tareas en las que están divididos.
  Estos usuarios tienen acceso a la información de todos los proyectos web (estado, horas presupuestadas, fecha inicio, fecha fin, etc.), creados por ellos mismos o por otros usuarios.
- Admin: este tipo de usuario es utilizado por los responsables de la empresa. Cuenta con la misma funcionalidad que "user" y además tiene el privilegio de gestionar los empleados y estadísticas. En particular, respecto a la gestión de estadísticas, un administrador puede comprobar tanto las horas invertidas en los proyectos cada día, como lo que ha realizado cada trabajador a diario y las horas que ha invertido.

El propósito de la empresa es mejorar la plataforma de gestión de proyectos ya existente en base a 3 objetivos fundamentales que se abordarán en este proyecto.

- En primer lugar, interesará solucionar algunos errores existentes que impiden el uso correcto de la aplicación.
- En segundo lugar, es necesario desarrollar una nueva interfaz para algunas páginas de la aplicación, mejorando la usabilidad de forma que los usuarios puedan gestionar la información de manera más sencilla.
- Por último, la empresa desea que se realicen mejoras puntuales para mejorar el funcionamiento de la aplicación.

1.2 Tecnologías

Debido a que se trata de un proyecto de reingeniería, para llevar a cabo los objetivos indicados anteriormente será necesario analizar la aplicación existente y estudiar las tecnologías utilizadas en el desarrollo de la misma. En particular, las tecnologías utilizadas son:

- *PHP 7.0,* como lenguaje de programación, junto con el Framework *Laravel* (versión 5.0). Este framework permite el uso de una sintaxis elegante y expresiva para crear código de forma sencilla. Como el servidor de la empresa en el que se desplegará finalmente el resultado del trabajo, está configurado con estas versiones y no se me permite actualizarlo a versiones posteriores, el presente proyecto utilizará las mismas versiones.
  Se han utilizado herramientas incluidas en *Laravel*, tanto para la gestión de *backend* como para el *frontend*.
    - Para el *backend* se utilizará el ORM de *Laravel* llamado *Eloquent*. Este framework nos permite manejar de una forma fácil y sencilla los procesos correspondientes al manejo de la información almacenada en bases de datos (BD). Además, *Eloquent* nos da la opción de realizar consultas completas utilizando para ello una sola línea de código.
    - Para el *frontend* se utilizarán las plantillas *Blade* de *Laravel*, las cuales nos permiten crear código HTML dinámico de forma más limpia que si usásemos *PHP puro*.
- Respecto a la base datos, se utilizará *MySQL* como sistema gestor de base de datos. Para la visualización y realización de operaciones en la BD se ha elegido la aplicación *PhpMyAdmin,* ya que proporciona una interfaz web intuitiva para trabajar con bases de datos *MySQL*.
- Para la maquetación de la aplicación se utilizará *HTML5, CSS3, JavaScript y el Framework Bootstrap*, el cual permite crear de forma sencilla aplicaciones que se adaptan a las distintas resoluciones (Web Responsive).
- Como IDE de desarrollo se ha elegido *PHPStorm* por dos principales razones. La primera de ellas es por ser muy fácil de configurar y permitir exportar e importar la configuración en un archivo. De esta manera se puede configurar *PHPStorm* de una manera sencilla, al disponer de una copia de mi configuración en la nube. La segunda razón es porque permite subir los archivos a un servidor con un solo click, teniendo una copia del proyecto en la nube, en todo momento.
- Además de todas estas tecnologías, también se usarán *Ajax, JQuery* y *JSON* para el desarrollo del proyecto.

1.3 Alcance

Teniendo en cuenta los 3 objetivos principales del presente Trabajo Fin de Grado, se ha decidido dividir el trabajo en 3 bloques, de los cuales el primero lo llevaré a cabo de forma independiente y autónoma (identificado en la subsección 1.3.1), mientras que los otros dos, los realizaré bajo la revisión de mi tutor en la empresa (identificados en las subsecciones 1.3.2, 1.3.3, respectivamente).

1.3.1 Estudio de las tecnologías y análisis del proyecto

Para la realización del proyecto será necesario estudiar el lenguaje de programación con el que está desarrollada la aplicación base. Por ello, principalmente se dedicará un tiempo para afianzar conocimientos de *PHP*, así como para el estudio del framework *Laravel*.

Tras el estudio de las tecnologías, se realizarán diversas pruebas de forma local para:

1) Asentar los conceptos de *Laravel*.
2) Conocer la estructura básica de un proyecto de *Laravel.*
3) Comprender la arquitectura *Modelo-Vista-Controlador*.

Seguidamente, analizaré el proyecto sobre el que realizaré este Trabajo Fin de Grado con el objetivo de comprender la estructura de los archivos que lo conforman.

Este estudio y análisis inicial me facilitará la tarea de averiguar el origen de los errores y realizar las mejoras referentes a la usabilidad y la funcionalidad de la aplicación.

1.3.2 Corrección de errores

Tras la primera reunión con el cliente, éste indicó que la aplicación tenía algunos fallos que no permitían usar correctamente alguna funcionalidad. Por ello, se dedicará una fase del presente proyecto a la corrección de los fallos señalados en dicha reunión.

Los fallos son los siguientes:

- Error en el módulo de las estadísticas de proyectos

  Este error tiene lugar al cambiar la fecha del selector de días para los cuales se desean obtener las estadísticas. Esto provoca que no se puedan visualizar las estadísticas de los proyectos de forma correcta.

- Error al ordenar los proyectos por fecha en el módulo de proyectos

  Existe un error que impide ordenar los proyectos por fecha de forma correcta.

- Error al eliminar proyectos en el módulo de proyectos

  Actualmente, la herramienta no permite eliminar proyectos.

1.3.3 Mejoras de usabilidad y rediseño de la interfaz

Tras hablar con el cliente, me comunicó varios puntos para este apartado, no obstante, los más importantes y los que mayor prioridad tienen, se encuentran en la página principal de la aplicación y en el módulo de proyectos. A continuación, se detallan brevemente.

- Mejoras en la página principal

Actualmente, los proyectos existentes se muestran en la página principal mediante un listado.

El cliente ha solicitado que el listado de proyectos pase a ser un campo de texto que sugiera coincidencias a medida que se vaya escribiendo en él.

Además, se añadirá un botón que despliegue una ventana emergente (pop-up) donde se pueda escribir una descripción de la tarea a realizar en cada proyecto.

Por último, se realizará una redistribución de la interfaz para que sea más clara e intuitiva.

- Mejoras en el módulo de proyectos.

El requisito con más interés para el cliente relacionado con este módulo, consiste en realizar una redistribución y mejora de la tabla donde se visualizan los proyectos. Actualmente, para modificar un proyecto, el director de proyectos debe pulsar el proyecto para entrar en la página de edición, donde debe buscar el dato que desea cambiar. Por ello, sería deseable que la tabla que visualiza los proyectos fuera editable. De esta forma se ahorraría mucho tiempo cada vez que se deseara actualizar un valor del proyecto, tarea que se realiza en la empresa con bastante frecuencia.

- Mejoras individuales

En esta parte se realizarán mejoras puntuales sobre la aplicación, y que no son específicas de ninguno de los módulos anteriores.

Una de ellas consistirá en desarrollar una página de error general para indicar de una forma clara qué está pasando y que permita regresar a un estado estable.

También se realizará una mejora en los filtros de búsqueda de las tablas de la aplicación, de manera que se permita buscar por varios criterios a la vez.

Tras estas mejoras, se comprobará que todas las funciones que envían correos electrónicos actúan de la forma esperada (este es el caso, por ejemplo, del botón de “recordar contraseña” o el de “invitar a usuario”, que deberían mandar un mensaje a la dirección de correo que indiquemos).

Por último, en el caso de disponer de tiempo, se realizarán otras mejoras como, eliminar la página inicial (pantalla con el logo de la empresa que redirige al login) de manera que se entre directamente al formulario de login, ahorrando tiempo al acceder a la aplicación. Además de lo anterior, el cliente nos ha sugerido la traducción de algunos apartados, el cambio de algunos títulos de tablas y colores, así como modificar la imagen de fondo por una más adecuada a la empresa.

De las mejoras citadas anteriormente, se intentará elegir aquellas que supongan algún tipo de reto o sean beneficiosas de alguna manera para el estudiante (como por ejemplo, que supongan el estudio puesta en práctica de alguna tecnología de interés para el estudiante).

Tras finalizar los tres bloques anteriores, realizaré un manual para la empresa. En este quedará explicada la nueva funcionalidad implementada.

1.4 Gestión del proyecto

Para la realización de este TFG dispongo de 300 horas, de las cuales la gran mayoría las pasaré en la empresa.

La fecha para la que tiene que estar acabado será la fecha del depósito (25-27 de junio), por ello, he decidido que voy a dedicar al TFG 5 horas diarias de lunes a viernes, llegando a un total de 25 horas semanales.

La fecha de inicio del TFG será el 19 de febrero de 2018, mientras que la fecha de finalización planeada será el 5 de junio de 2018, de manera que dejaré un margen entre la finalización y el depósito, por cualquier problema o contratiempo que pudiese surgir.

Además, al dedicar 5 horas diarias, me da la oportunidad de dejar algún día libre, si por alguna razón me hiciese falta (Semana Santa, asuntos universitarios o asuntos personales).

Por último, quiero indicar que compaginaré la realización del TFG con tres asignaturas, por lo que, en algún momento puntual, podría darse el caso de no poder realizar las 5 horas diarias comentadas anteriormente.

1.4.1 Metodología

Ya que las mejoras a realizar en la aplicación serán varias y de diferente naturaleza, la metodología de trabajo elegida ha sido la incremental. De esta manera, cada bloque se dividirá principalmente en 4 fases.

1. Análisis: Donde analizaré el código ya existente.
2. Diseño: Donde llevaré a cabo el diseño de una solución al problema planteado y/o realización de la toma de decisiones más adecuadas.
3. Implementación: En el que pondré en marcha del diseño realizado en la fase anterior.
4. Pruebas: Donde realizaré las pruebas finales para comprobar que la solución dada al problema funciona correctamente.

1.4.2 Estructura de descomposición de tareas (EDT)

Antes de realizar cualquier estimación, es necesario conocer las tareas en las que se divide el proyecto. En la siguiente figura se muestra el EDT del Trabajo Fin de Grado.

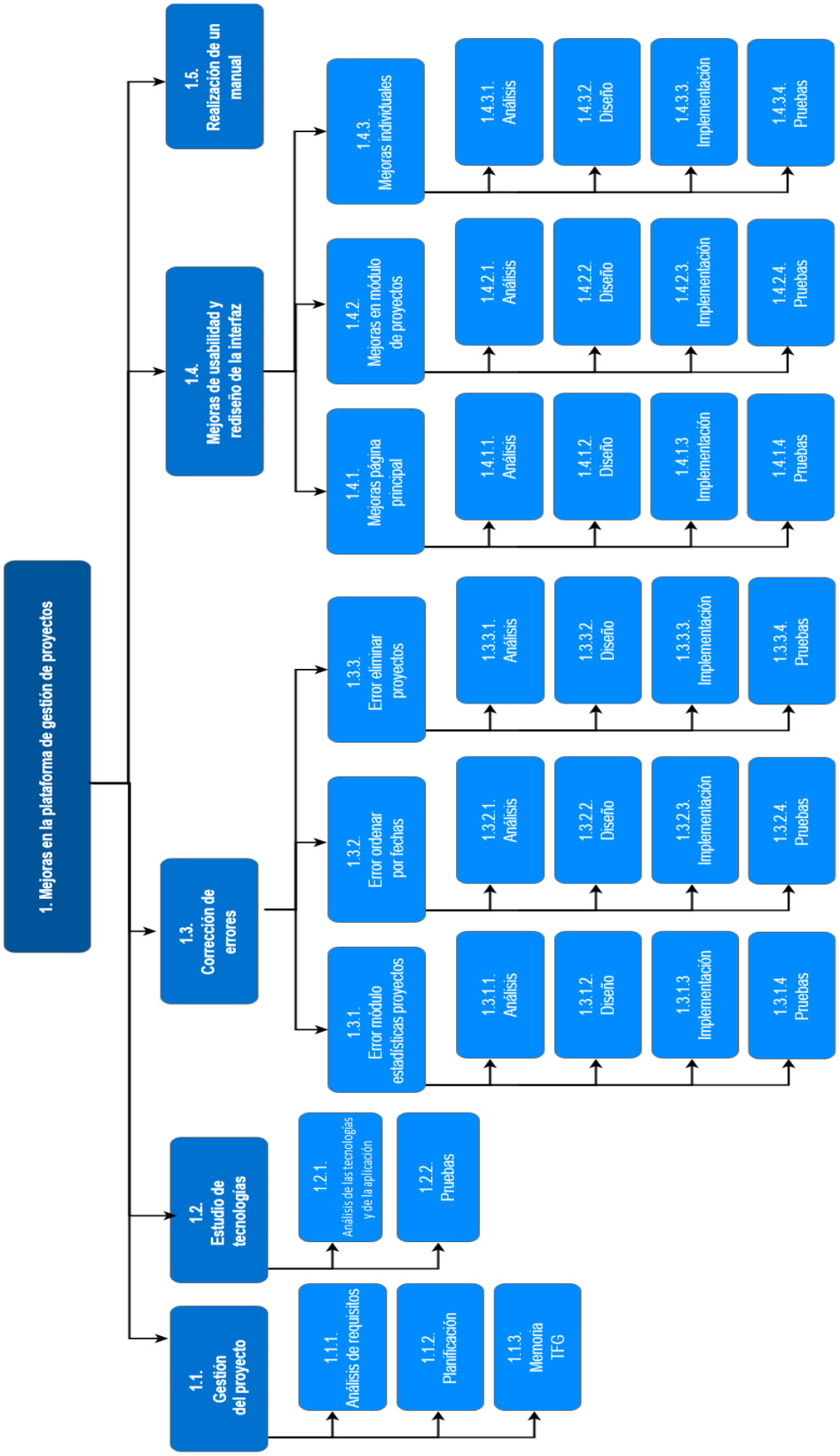


Figura 1.1 Estructura de descomposición de tareas

1.4.3 Descripción de tareas

En la siguiente tabla podemos ver de una forma ordenada y detallada todas las tareas a realizar, junto con la estimación de tiempo que dedicaremos a ellas.

Mejoras en la plataforma de gestión de proyectos			
Núm.	**Nombre**	**Descripción**	**Horas estimadas**
1.1	**Planificación del proyecto**		**81 horas**
1.1.1.	Análisis de requisitos	Reunión inicial con el tutor de la empresa para mostrarme el funcionamiento de la aplicación y describirme, exponerme los requisitos, además de ordenarlos por bloques.	5 horas
1.1.2.	Planificación	Definir la metodología a utilizar, realización del grafico EDT, diagrama de Gantt e hitos, plan de comunicación, contingencia y calidad.	16 horas
1.1.3.	Memoria TFG	Realización de la memoria del TFG.	60 horas
1.2.	**Estudio de tecnologías**		**24 horas**
1.2.1.	Análisis	Análisis y aprendizaje de las tecnologías a utilizar, en mayor parte *Laravel* y la estructura del proyecto.	18 horas
1.2.2.	Prueba	Instalación y montaje de la aplicación en un entorno local.	6 horas
1.3.	**Corrección de errores**		**25 horas**
1.3.1.	Error en módulo de estadísticas	Error al cambiar de fecha dentro del selector.	5 horas
1.3.1.1.	Análisis	Análisis del error en la aplicación y del código existente.	0.5 horas
1.3.1.2.	Diseño	Buscar la solución más adecuada para el problema encontrado.	1 hora
1.3.1.3.	Implementación	Implementación la solución encontrada anteriormente.	3 horas
1.3.1.4.	Pruebas	Comprobar que el problema se ha resuelto y todo funciona correctamente.	0.5 horas
1.3.2.	Error ordenar por fechas	Error al ordenar los proyectos por fecha.	5 horas
1.3.2.1.	Análisis	Similar a 1.3.1.1.	0.5 horas
1.3.2.2.	Diseño	Similar a 1.3.1.2.	1 hora
1.3.2.3.	Implementación	Similar a 1.3.1.3.	3 horas
1.3.2.4.	Pruebas	Similar a 1.3.1.4.	0.5 horas
1.3.3.	Error al eliminar proyectos	Error que no permite eliminar proyectos.	15 horas
1.3.3.1.	Análisis	Similar a 1.3.1.1.	1 hora
1.3.3.2.	Diseño	Similar a 1.3.1.2.	2 horas
1.3.3.3.	Implementación	Similar a 1.3.1.3.	10 horas
1.3.3.4.	Pruebas	Similar a 1.3.1.4.	2 horas

1.4.	**Mejoras de usabilidad y rediseño de la interfaz**		**165 horas**
1.4.1	Mejoras página principal		47 horas
1.4.1.1.	Análisis	Analizar los requisitos del cliente a realizar y revisión del código existente que afecta a dichos requisitos.	5 horas
1.4.1.2.	Diseño	Diseño de la interfaz de la página principal de la aplicación. En caso de ser necesario, se modificará la base de datos, se diseñarán varias soluciones y se elegirá la más adecuada.	5 horas
1.4.1.3.	Implementación	Realizar la interfaz diseñada anteriormente y hacer los cambios en la base de datos, en caso de ser necesario.	35 horas
1.4.1.4.	Pruebas	Comprobar que el funcionamiento es el esperado y que todo funciona correctamente.	2 horas
1.4.2	Mejoras en el módulo de proyectos		68 horas
1.4.2.1.	Análisis	Similar a 1.4.1.1.	5 horas
1.4.2.2.	Diseño	Similar a 1.4.1.2.	5 horas
1.4.2.3.	Implementación	Similar a 1.4.1.3.	55 horas
1.4.2.4.	Pruebas	Similar a 1.4.1.4.	3 horas
1.4.3	Mejoras individuales		50 horas
1.4.3.1.	Análisis	Similar a 1.4.1.1.	5 horas
1.4.3.2.	Diseño	Similar a 1.4.1.2.	5 horas
1.4.3.3.	Implementación	Similar a 1.4.1.3.	35 horas
1.4.3.4.	Pruebas	Similar a 1.4.1.4.	5 horas
1.5.	**Realización de un manual**	Realización de un manual sobre cómo utilizar las nuevas funcionalidades implementadas.	**5 horas**
		Total	**300 horas**

1.4.4 Diagrama de hitos

En la Figura 1.2 podemos ver las fechas para las cuales está prevista la finalización de cada tarea, así como las fechas de las reuniones con ambos tutores.

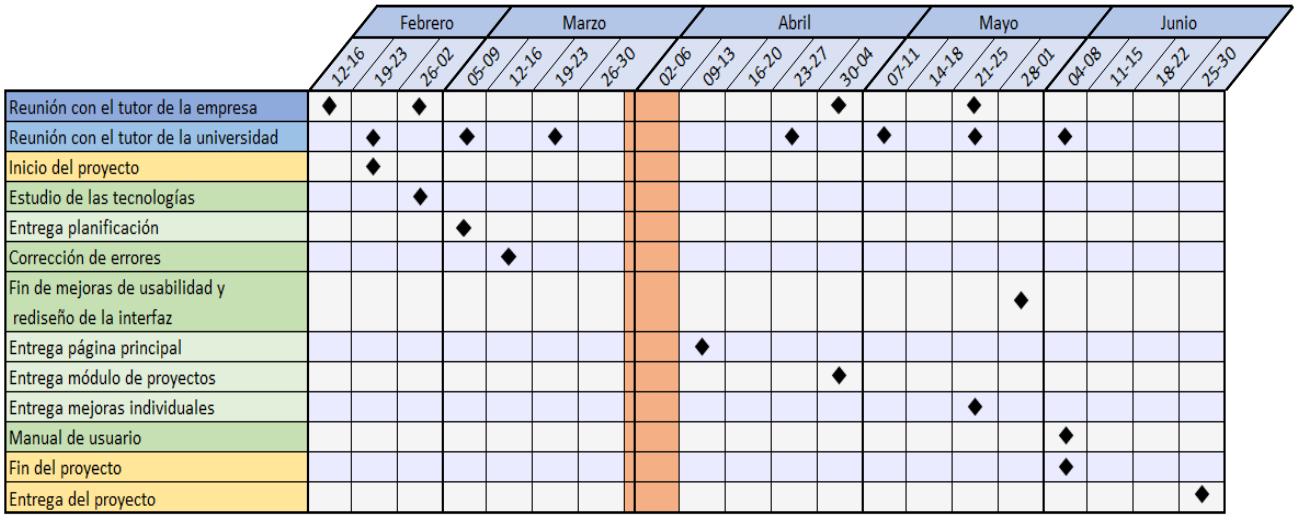


Figura 1.2 Diagrama de hitos

1.4.5 Diagrama de Gantt

En las siguientes tablas podemos ver cómo he previsto el tiempo que dedicaré a cada tarea.

			Febrero										Marzo															Abril												
			Semana 1					Semana 2					Semana 3					Semana 4					Semana 5					Semana 6					Semana 7					Semana 8				
			19	20	21	22	23	26	27	28	1	2	5	6	7	8	9	12	13	14	15	16	19	20	21	22	23	26	27	28	29	30	2	3	4	5	6	9	10	11	12	13
1.1	**Gestión del proyecto**	**81 horas**																																							
1.1.1.	Análisis de requisitos	5 horas																																							
1.1.2.	Planificación	16 horas																																							
1.1.3.	Memoria TFG	60 horas																																							
1.2.	**Estudio de tecnologías**	**24 horas**																																							
1.2.1.	Análisis	18 horas																																							
1.2.2.	Prueba	6 horas																																							
1.3.	**Corrección de errores**	**25 horas**																																							
1.3.1.	Error en módulo de estadísticas	5 horas																																							
1.3.2.	Error ordenar por fechas	5 horas																																							
1.3.3.	Error al eliminar proyectos	15 horas																																							
1.4.	**Mejoras de usabilidad y rediseño de la interfaz**	**165 horas**																																							
1.4.1	Mejoras página principal	47 horas																																							
1.4.1.1.	Análisis	5 horas																																							
1.4.1.2.	Diseño	5 horas																																							
1.4.1.3.	Implementación	35 horas																																							
1.4.1.4.	Pruebas	2 horas																																							
1.4.2	Mejoras en el módulo de proyectos	68 horas																																							
1.4.2.1.	Análisis	5 horas																																							
1.4.2.2.	Diseño	5 horas																																							
1.4.2.3.	Implementación	55 horas																																							
1.4.2.4.	Pruebas	3 horas																																							
1.4.3	Mejoras individuales	50 horas																																							
1.4.3.1.	Análisis	5 horas																																							
1.4.3.2.	Diseño	5 horas																																							
1.4.3.3.	Implementación	35 horas																																							
1.4.3.4.	Pruebas	5 horas																																							
1.5.	**Realización de un manual**	**5 horas**																																							

Tabla 1.3 Diagrama de Gantt de la semana 1 a la 8

			Abril										Mayo																						Junio						
			Semana 9					Semana 10					Semana 11					Semana 12					Semana 13					Semana 14					Semana 15					Semana 16				
			16	17	18	19	20	23	24	25	26	27	30	1	2	3	4	7	8	9	10	11	14	15	16	17	18	21	22	23	24	25	28	29	30	31	1	4	5	6	7	8
1.1	**Gestión del proyecto**	**81 horas**																																							
1.1.1.	Análisis de requisitos	5 horas																																							
1.1.2.	Planificación	16 horas																																							
1.1.3.	Memoria TFG	60 horas																																							
1.2.	**Estudio de tecnologías**	**24 horas**																																							
1.2.1.	Análisis	18 horas																																							
1.2.2.	Prueba	6 horas																																							
1.3.	**Corrección de errores**	**25 horas**																																							
1.3.1.	Error en módulo de estadísticas	5 horas																																							
1.3.2.	Error ordenar por fechas	5 horas																																							
1.3.3.	Error al eliminar proyectos	15 horas																																							
1.4.	**Mejoras de usabilidad y rediseño de la interfaz**	**165 horas**																																							
1.4.1	Mejoras página principal	47 horas																																							
1.4.1.1.	Análisis	5 horas																																							
1.4.1.2.	Diseño	5 horas																																							
1.4.1.3.	Implementación	35 horas																																							
1.4.1.4.	Pruebas	2 horas																																							
1.4.2	Mejoras en el módulo de proyectos	68 horas																																							
1.4.2.1.	Análisis	5 horas																																							
1.4.2.2.	Diseño	5 horas																																							
1.4.2.3.	Implementación	55 horas																																							
1.4.2.4.	Pruebas	3 horas																																							
1.4.3	Mejoras individuales	50 horas																																							
1.4.3.1.	Análisis	5 horas																																							
1.4.3.2.	Diseño	5 horas																																							
1.4.3.3.	Implementación	35 horas																																							
1.4.3.4.	Pruebas	5 horas																																							
1.5.	**Realización de un manual**	**5 horas**																																							

Tabla 1.4 Diagrama de Gantt de la semana 9 a la 16

1.4.6 Plan de comunicación.

Durante el periodo de realización del TFG se realizarán reuniones periódicas tanto con el tutor de la empresa como con el tutor de la Universidad. La forma en la que se realizarán dichas comunicaciones será:

- Comunicación con el tutor de la empresa:
    - Para cualquier duda que tenga lugar dentro del horario de la realización del TFG dentro de la empresa, la comunicación se hará de forma oral, ya que es la forma más rápida de solucionar cualquier problema.
    - Para cualquier duda que tenga lugar fuera de dicho horario, la comunicación se realizará mediante el correo electrónico que me ha proporcionado la empresa.
- Comunicación con el tutor de la Universidad: se realizará con mi correo personal de la Universidad de La Rioja. Además, sostendré una comunicación de forma semanal, con el fin de mantenerle informado de mis avances.

1.4.7 Plan de contingencia y calidad

Para la realización del presente trabajo, seguiré los siguientes planes de contingencia y calidad.

1.4.7.1 Contingencia

Para evitar cualquier tipo de problemas relacionados con la pérdida de datos se mantendrán todos los datos tanto en el ordenador donde realizo el TFG, como en la nube (*OneDrive* proporcionado por la Universidad).

En el caso de que el cliente no esté disponible, no se pasará de fase hasta que éste dé el visto bueno de lo realizado hasta el momento. Mientras tanto, me dedicaré a realizar la memoria.

Si el cliente pide ciertas ampliaciones sobre una fase que ya está cerrada, estas serán valoradas con objeto de estimar el coste de tiempo necesario para realizarlas. En caso de ser aceptadas, el tiempo que se invierta en su realización será descontado de la fase del proyecto dedicado a mejoras individuales.

Como último problema, en el caso de no ser posible realizar el TFG temporalmente, ya sea por enfermedad o por causas personales, se utilizará el tiempo dedicado en la planificación a estos casos.

1.4.7.2 Calidad

Para comprobar el correcto funcionamiento de la aplicación, tras acabar cada fase, se efectuará una reunión con el tutor de la empresa de manera que pueda valorar el trabajo realizado y comprobar que lo desarrollado era lo esperado.

2. Fase 1. Estudio de la aplicación y las tecnologías.

En esta fase se analizará la aplicación existente con un grado de profundidad suficiente como para tener una idea de la misma. Asimismo, estudiaremos las diferentes tecnologías que se utilizarán en el TFG. En particular, como se ha comentado anteriormente, nos familiarizaremos con las tecnologías, con objeto de comprenderlas con suficiente profundidad como para poder entender el funcionamiento interno de la aplicación.

2.1 Análisis de las tecnologías

Tras un primer vistazo a la aplicación ya existente, se ha podido comprobar que todas las tecnologías (*HTML5, CSS3, JQuery, JavaScript, Bootstrap, Ajax*) ya habían sido utilizadas (algunas de ellas no con mucha profundidad) durante la realización de las prácticas llevadas a cabo por el estudiante en la misma empresa durante el primer semestre.

Merece la pena destacar el uso de *Laravel* para el desarrollo de la herramienta. Este framework fue utilizado de una forma superficial durante la realización de las prácticas, por lo que no se dispone de un dominio idóneo como para tener suficiente solvencia para poder empezar con este proyecto. Por ello, antes de comenzar con este proyecto, he necesitado documentarme para poder comprender determinadas partes de la aplicación. Como una tecnología no puede ser dominada completamente en tan poco tiempo, decidí estudiar lo necesario como para comprender el funcionamiento básico de la aplicación y, más adelante, en cada incremento, estudiar las partes de la tecnología que hicieran falta, con objeto de cumplir con los requisitos del cliente.

2.2 Estructura de la aplicación

Como se ha comentado anteriormente, la aplicación se desarrolló con el Framework de *PHP* llamado *Laravel* en su versión 5.0*.,* siguiendo el patrón MVC (Modelo Vista Controlador). Esto ha requerido la realización de un pequeño repaso para comprender la estructura de carpetas que conforman la aplicación.

En particular, las *vistas* se encuentran en la carpeta `resources/views`, los *controladores* se encuentran en la carpeta `Http/Controllers` *y* los *modelos* dentro de la carpeta `app`. En la Figura 2.1 se pueden observar tres columnas:

- En la primera columna se ven los elementos del *modelo* de negocio de la aplicación como por ejemplo *`Empresa`*, *`Modulo`*, *`Proyecto`* *y* *`Trabajador`*.
- En la segunda columna podemos ver las *vistas*.
- En la última columna se pueden observar los *controladores* que tienen un nombre descriptivo para saber con qué modelo de negocio están relacionados.

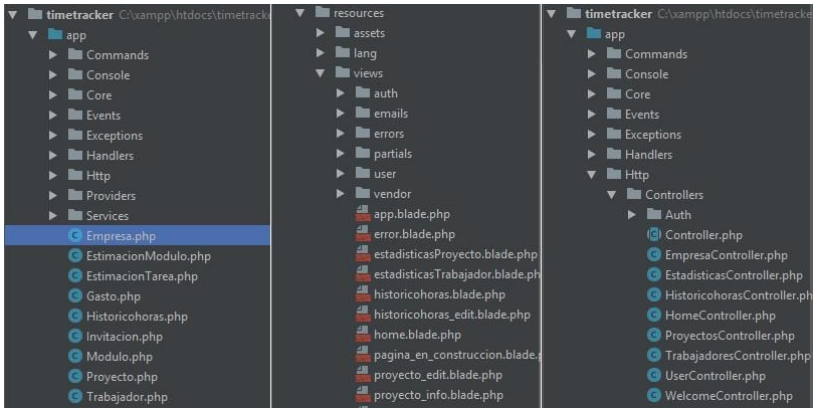


Figura 2.1 Estructura del proyecto

Además, he analizado el funcionamiento de uno de los ficheros más importantes dentro de un proyecto *Laravel*, el archivo `Routes.php`, en el cual se definen todas las URLS junto con el tipo de petición que procesarán y a qué función del controlador corresponden. En la Figura 2.2 se puede ver un ejemplo de ruta y las partes por las que está formada.

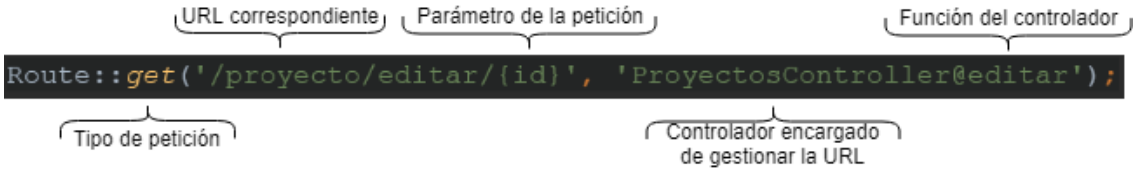


Figura 2.2 Análisis de una Ruta Laravel

También hay que destacar que he tenido que dedicar un pequeño tiempo a comprender la estructura utilizada por las plantillas *Blade* (nombre que se le da a las vistas dentro del framework *Laravel*) de esta aplicación. Como se puede ver en la siguiente Figura 2.3, dichas plantillas están formadas por secciones, como por ejemplo `app` (bloque *HTML* que contiene la barra de navegación, ver la línea 1), `css` (bloque donde se importarán las hojas de estilo, ver las líneas de la 11 a 14) y `content` (sección donde se escribirá el código HTML correspondiente a la vista, ver las líneas de la 16 a la 18).

```
1  @extends('app')
2
3  @section('title')
4      Timetracker - Home
5  @stop
6
7  @section('description')
8      Introducir las horas trabajadas en la empresa.
9  @stop
10
11 @section('css')
12     <link href="{{ asset('/css/adicionalHome.css') }}" rel="stylesheet">
13
14 @stop
15
16 @section('content')
17
18 @endsection
19
20 @section('javascript')
21     <script src="https://cdnjs.cloudflare.com/ajax/libs/typeahead.js/0.11.1/typeahead.bundle.min.js"></script>
22 @endsection
```

Figura 2.3 Estructura de las plantillas Blade

Seguidamente, y debido a que se trata de un proyecto de reingeniería, me gustaría destacar uno de los puntos problemáticos con los que me he encontrado y es que todo el código de la aplicación estaba sin documentar. Este hecho ha sido el causante de que haya necesitado un mayor tiempo para realizar los requisitos de este TFG, resultando todo un reto cada vez que tenía que realizar cualquier mejora o encontrar los errores señalados por el cliente.

3. Fase 2. Corrección de errores

En esta fase se tratará de corregir una serie de fallos que contiene la aplicación actual y que impiden al cliente el uso correcto de la aplicación. Es por ello que esta fase se realiza antes de comenzar a mejorar la usabilidad de la interfaz y la funcionalidad.

3.1 Error en el módulo de estadísticas

Como se ha adelantado previamente, este error ocurre a la hora de intentar visualizar las estadísticas de un proyecto entre dos fechas.

Para ver de dónde provenía este error, se comprobó desde el inspector del navegador que la petición *Ajax* estaba dando un error 500 (como se puede ver en la Figura 3.1).

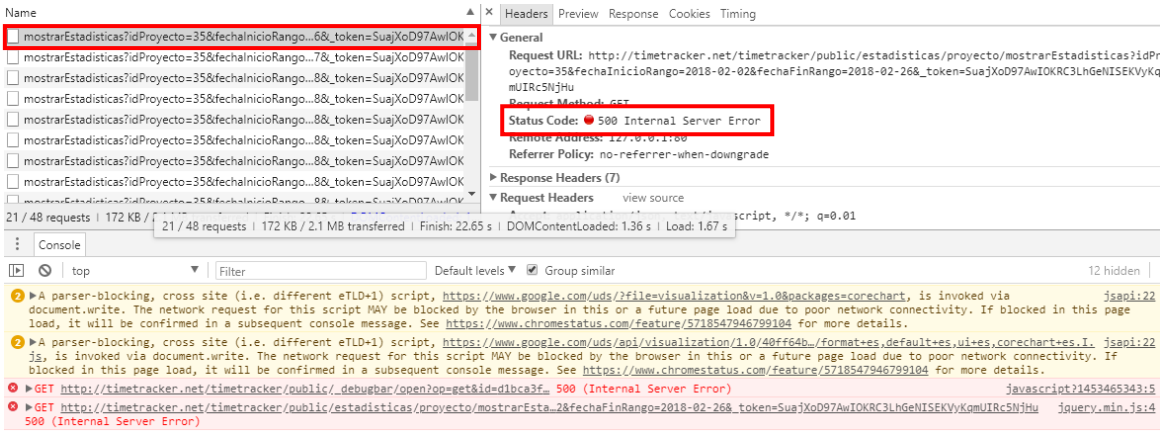


Figura 3.1 Error en la petición Ajax

Para comprobar exactamente el motivo del error, se utilizó la barra de debug que nos proporciona *Laravel.* Mediante el debug se pudo observar que el error se debía a un intento de división por 0 en la función `getDiasLaborablesEntreFechas`.

Tras encontrar el error, la solución que se tomó fue volver a desarrollar la función entera, ya que lo que debería haberse desarrollado con unas simples líneas (ver Figura 3.2), la versión antigua de dicho método era innecesariamente compleja, dedicando muchas líneas de código. En particular, en la Figura 3.2 se muestra la instrucción que compara que el día que se le pasa por parametro a la función no es ni 6 ni 7 (sabado o domingo), para así sumar una unidad al contador de días hábiles.

Por último se realizarán las pruebas correspondientes para confirmar que el funcionamiento era el adecuado.

```
if (!in_array(date('N', $i), array(6,7))) {
    // Aumento contador
    $diashabiles++;
}
```

Figura 3.2 Comprobación en `getDiasLaborablesEntreFechas`

3.2 Error al ordenar los proyectos por fecha

Este error tenía lugar dentro del módulo de proyectos, al intentar ordenar la tabla que los visualiza por fecha.

Tras analizar el código existente y ver su funcionamiento, pude comprobar que el problema radicaba en el código de la función `sort_ddmm`, que se encarga de ordenar los proyectos por fecha. En particular, esta función usaba el operador >, no obstante, la aplicación no utilizaba el formato de fecha requerido por dicho operador.

Para solucionarlo decidí desarrollar una nueva función `sort_Fechas`, de forma que me permitiese comparar fechas con el formato que usa la aplicación. Para ello hice una copia de la función `sort_ddmm` y modifiqué las líneas de código necesarias. Tras esto, cambié las líneas de código donde se invocaba `sort_ddmm`, por `sort_Fechas`.

Además, para poder ordenar la tabla por fechas, tenía el inconveniente de que había proyectos con fecha nula y éstos no se podían ordenar. En particular, había proyectos en la tabla `proyectos` que no tenían fecha de finalización, sino NULL, lo que nos impedía que la tabla se pudiera ordenar correctamente (ya que no puede comparar fechas nulas). Por ello, modifiqué los proyectos con fecha nula indicando como fecha fin `'0000-00-00'`. Además, actualicé el campo `` `fechafin` `` para que por defecto cuando se introduzca un nuevo el campo tenga el valor`'0000-00-00'`.

3.3 Error al eliminar proyectos

Este era uno de los mayores problemas para el cliente, ya que corresponde a una de las funcionalidades principales de la aplicación, de ahí su interés en corregirlo.

3.3.1 Análisis

Los usuarios "admin" disponen de la opción de eliminar proyectos. El borrado de un proyecto requiere la eliminación de registros de varias tablas. Esta funcionalidad no está disponible actualmente en la aplicación, devolviendo un error causado porque la función utilizada para eliminar proyectos todavía no está implementada.

En la empresa se me plantearon dos opciones para hacer el borrado de los proyectos:

- Borrado físico: donde la información correspondiente al proyecto es borrada totalmente de la base de datos.
- Borrado lógico: donde la información no se borra, sino que simplemente deja de listarse de una forma pública. Para ello se utilizaría un campo en la base de datos que indique el estado de cada proyecto (“eliminado” o no).

3.3.2 Diseño

Tras analizar las dos opciones anteriores con mi tutor de la empresa, elegimos un borrado lógico, ya que estaba interesado en disponer de un listado de proyectos limpio, es decir, solo se listarán los proyectos en los que se está trabajando. Además, de esta forma se mantiene la información de los proyectos cerrados o antiguos en la base de datos, dando la posibilidad de poder continuar con ellos en un futuro.

Seguidamente, revisé el código de la aplicación comprobando finalmente que el borrado del proyecto se realiza a través de un método `borrar`, el cual se encuentra dentro del controlador de los proyectos (*`ProyectosController.php`*) como se puede ver en la Figura 3.4.

Figura 3.4 Función `borrar` en `ProyectosController.php`

Como se puede observar en la Figura 3.4, en la línea 4 aparece comentada la línea de código que invocaría al método `eliminar`, el cual está sin implementar.

Para llevar a cabo la estrategia basada en un borrado lógico, analicé la base de datos y observé que era necesario crear un campo nuevo en la tabla `proyectos`, el cual permita comprobar el estado en el que se encuentra cada proyecto de la empresa. Además, sería necesario actualizar las consultas que listarán todos los proyectos para que no se muestren los proyectos eliminados.

Por último, habrá que modificar los controladores necesarios para que cuando se le pase una URL con un proyecto eliminado, devuelva el error correspondiente y no permita acceder a su información.

3.3.3 Implementación

En lo que se refiere a la BD, como se ha comentado anteriormente, tuve que crear un nuevo campo en la tabla `proyectos`. Hemos llamado a dicho campo de una forma significativa, con el nombre de `eliminado`, de tipo `booleano` y por defecto `falso`.

Tras esto, se ha desarrollado finalmente la función `eliminarLogico` para poder eliminar un proyecto de manera lógica. Con objeto de respetar el patrón MVC, se ha desarrollado esta función dentro del modelo *Proyecto* (*`Proyectos.php`*).

La función `eliminarLogico` (ver Figura 3.5) se encargará de comprobar el estado del proyecto seleccionado y, en caso de no estar “eliminado”, se actualizará el proyecto seleccionado a un estado "eliminado". Este método devolverá un booleano en función de sí el proyecto se ha eliminado o no correctamente.

```php
//Elimina sin llegar a borrar de la BD
public function eliminarLogico()
{
    $estado = Proyecto::select("eliminado")
                ->where('id','=',$this->id)
                ->pluck('eliminado');
    if($estado==0){
       return Proyecto::where('id','=',$this->id)
            ->update(['eliminado' => 1]);
    }else{
        return 0;
    }
```

Figura 3.5 Función `eliminarLogico` dentro del modelo `Proyecto`

Posteriormente, he modificado la función `borrar` del controlador (ver Figura 3.4) de manera que invoque al nuevo método `eliminarLogico`.

La siguiente tarea ha consitido en actualizar la vista encargada de listar todos los proyectos, con objeto de evitar mostrar los que estén eliminados. Para ello he recurrido a la plantilla *Blade* llamada *`proyectos`* y he añadido una comprobación para evitar listar los proyectos eliminados (ver recuadro resaltado en la Figura 3.6).

```
<tbody>
    @for($i=0 ; $i<count($proyectosConHorasReales); $i++)
    @if($proyectosConHorasReales[$i][0]->eliminado!=1)
        @if(!empty($proyectosConHorasReales[$i][0]->fechafin))
            @if((strtolower($proyectosConHorasReales[$i][0]->estado) == 'desarrollo') && (date('Y-m-d')
            <tr class="danger">
            @elseif((strtolower($proyectosConHorasReales[$i][0]->estado) == 'desarrollo') && ($proyecto
            <tr class="warning">
```

Figura 3.6 Comprobación en la plantilla Blade Proyectos

Por último, se han llevado a cabo las comprobaciones necesarias para no poder acceder mediante una URL a un proyecto que está "eliminado".

Para ello, se ha analizado tanto el controlador de proyectos `proyectoController.php` (concretamente la función de *`editar`*), como el controlador de estadísticas `estadisticasController.php` (concretamente la función *`indexProyectos`*), de forma que antes de redirigir a la vista correspondiente comprueben el estado del proyecto y, en caso de que esté "eliminado", la aplicación redirija a la vista de error con el mensaje adecuado (ver Figura 3.7).

```
71          $proyecto = Proyecto::find($id);
72
73          if($proyecto->eliminado!=0 || $proyecto==null){
74              return view('error')->with('message', 'El proyecto no existe');
75          }
76          ...
```

Figura 3.7 Comprobación en los controladores

3.3.4 Pruebas

Tras realizar la fase de implementación, he observado que los proyectos se eliminan correctamente. Además, también he comprobado que no se puede acceder ni editar mediante la aplicación un proyecto que esté en un estado "eliminado", redirigiéndonos a una página de error donde nos indica que dicho proyecto no existe.

4. Fase 3. Mejoras en la página principal

En esta fase abordaremos todas las mejoras propuestas por el cliente para el módulo de la página principal.

4.1 Análisis

Para comenzar, debo decir que este módulo es utilizado principalmente por cada trabajador, ya que registra diariamente en él sus horas trabajadas. Esta es una de las principales razones de la mejora, ya que el cliente está interesado en facilitar lo máximo posible esta tarea a los trabajadores.

En particular, los principales requisitos del cliente que atañen a este módulo son:

- En primer lugar, sustituir el listado (`comboBox`) que contiene todos los nombres de los proyectos por un campo que dé sugerencias de proyectos a medida que escribimos en él.
- Segundo, implementar la funcionalidad necesaria para que las tareas introducidas por cada empleado puedan tener una descripción, de manera que cuando el administrador las revise, le resulte más sencillo saber en qué tareas han invertido las horas sus trabajadores.
- Por último, se modificará el código HTML y CSS para darle un estilo más actual y similar al de la aplicación web. Además, también realizaré unas pequeñas mejoras como colocar la fecha en formato extendido del día en que estamos, actualizar el icono de la "mano" que aparece en el campo de las horas asociadas a cada proyecto y actualizar el estilo de los botones de este módulo.

4.2 Diseño

Para abordar estos requisitos, los pasos que he tomado han sido los siguientes:

- Para la parte del campo con sugerencias, he encontrado una librería de *JavaScript* llamada *TypeaHead* [1] que, configurada adecuadamente, cumple a la perfección con los requisitos del cliente, el cual quiere algo similar al boceto que mostramos a continuación:

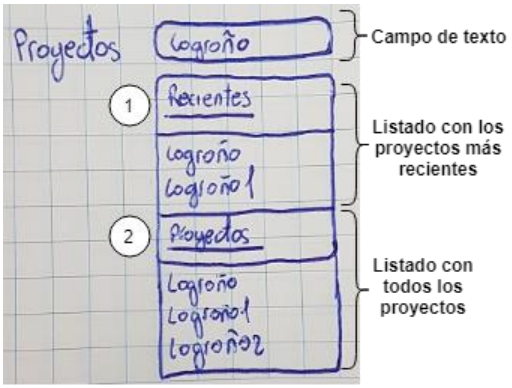


Figura 4.1 Boceto del campo de sugerencias.

Como se puede apreciar en la imagen anterior, a medida que el usuario vaya escribiendo en el campo de texto, se mostrarán dos listados, uno con los proyectos más recientes y otro con todos los proyectos.

- Respecto a la descripción de las tareas, he decidido que reajustaré ligeramente la tabla en la que se muestran las tareas de forma que quepa un nuevo icono, de manera que, al pulsarlo, despliegue un pop-up donde se podrá introducir una descripción. De esta manera la tabla seguirá teniendo un aspecto muy similar a la actual y además tendrá la nueva funcionalidad.
- Para la parte del HTML, el único requisito del cliente es que aparezca, en algún lugar de la pantalla, la fecha en formato extendido en la cual se están introduciendo las horas dedicadas a un proyecto. Respecto al CSS, se hará un retoque a la vista completa, centrándonos especialmente en actualizar la parte de la tabla donde se muestran las horas estimadas, realizadas y la diferencia de horas.

4.3 Implementación

A continuación, se presentan los aspectos de más relevancia referentes a la implementación de esta fase.

4.3.1 Creación de un campo de texto con sugerencias

Para comenzar, lo primero que hice fue leer todas las configuraciones del plugin de *jQuery*, con objeto de encontrar la más adecuada para los requisitos del cliente.

Tras esto, desarrollé una función llamada `callTypeahead`, la cual se ejecutará en el momento en el que se cargue la página, y que se ocupará de realizar todo el proceso que conlleva el gestionar lo que el usuario va escribiendo en el campo de búsqueda, así como de realizar las correspondientes sugerencias. Para explicar el proceso se ha diseñado la Figura 4.2, en la que se muestran 3 pasos fundamentales.

En primer lugar, se ha elegido una configuración que permite obtener los resultados de las sugerencias de una forma remota (ver líneas 811 y 821 del punto 1). En particular, cada vez que el usuario pulse una tecla, el plugin hará una llamada *Ajax* a una función del controlador (ver línea 282 del punto 3), la cual será elegida mediante la ruta correspondiente (ver punto 2). Dicha llamada *Ajax* hará una consulta a la base de datos, devolviendo el resultado de la misma en formato *JSON* (ver línea 288 del punto 3). Finalmente, la función `callTypeahead` gestionará dicho JSON mostrando el contenido en el campo del listado.

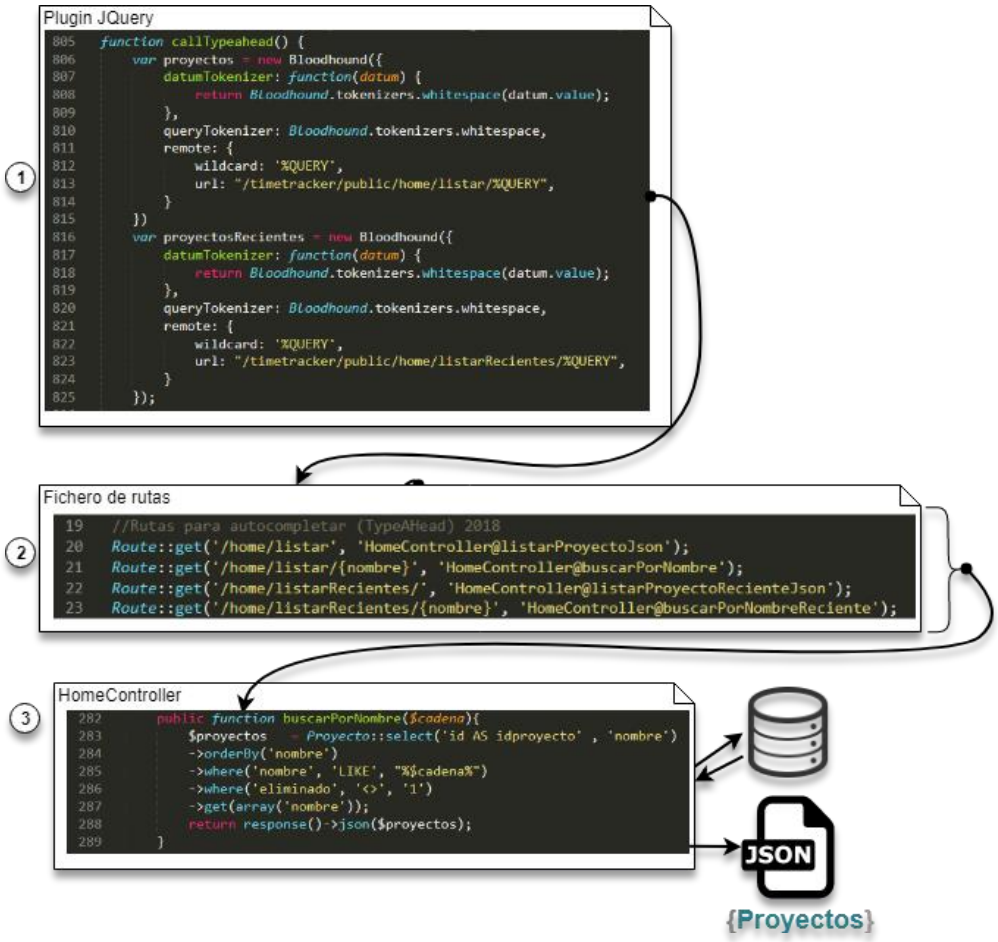


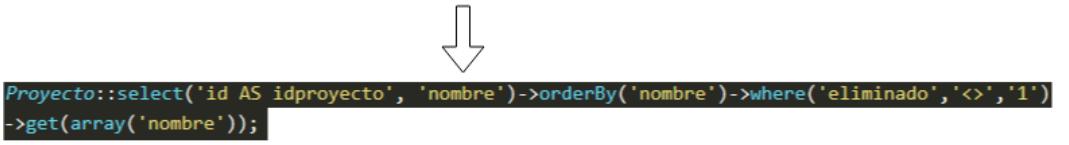
Figura 4.2 Diagrama de funcionamiento del método `callTypeahead`.

Además, como se puede ver en las líneas 806 y 816 del punto 1, he desarrollado esta función de forma que crea dos listados de proyectos, con el fin de que el campo con las sugerencias muestre ambos. El primero de ellos listará los proyectos más recientes (ver líneas de la 816 a la 825 del punto 1, que a su vez utilizan las URLs de las líneas 22 y 23 del punto 2). Por otro lado, el segundo mostrará todos los proyectos de la empresa (ver líneas de la 806 a la 815 del punto 1, que a su vez utilizan las URLs de las líneas 20 y 21 del punto 2).

En particular, para mostrar los dos tipos de proyectos se han implementado 4 funciones en el controlador (`buscarPorNombre` y `listarProyectoJson`, por un lado, y `listarProyectoRecienteJson` y `buscarPorNombreReciente`, por otro). Dichas funciones se invocan desde la rutas del controlador que se pueden ver en las líneas 813 y 823 del punto 1. Para desarrollar dichas funciones he seguido los siguientes pasos.

- Funciones para listar todos los proyectos: `buscarPorNombre` y `listarProyectoJson`. Tras leer la documentación sobre el *ORM Eloquent* de *Laravel*, estas dos funciones me resultaron bastante fáciles de implementar, ya que corresponden a consultas muy simples y, usando el *ORM,* se podían definir con una sola línea, como se puede ver en la Figura 4.3.

```
SELECT nombre AS idproyecto FROM Proyecto WHERE eliminado <> 1 ORDER BY nombre
```



```
Proyecto::select('id AS idproyecto', 'nombre')->orderBy('nombre')->where('eliminado','<>','1')
->get(array('nombre'));
```

Figura 4.3 Representación de una consulta SQL utilizando el ORM de Laravel

- Funciones para listar los proyectos recientes: `listarProyectoRecienteJson` y `buscarPorNombreReciente`. Estas funciones listarán en particular los 3 últimos proyectos en los que ha trabajado el propio trabajador que tiene la sesión activa. En particular, en la implementación de las consultas incluidas en dichas funciones, surgió el siguiente problema. En el momento de crear la consulta base, como ambas funciones utilizan la misma consulta a diferencia de un LIKE (se puede ver en el recuadro verde de la Figura 4.4), la primera idea que tuve fue hacer una consulta con un simple JOIN de dos tablas (`historicoHoras` y `proyectos`), un ORDER BY para ordenar por fecha y por último, eliminar los valores repetidos mediante un DISTINCT. No obstante, tras hacer las comprobaciones necesarias, observé que esta consulta ejecutaba como última orden el DISTINCT y por ello los valores devueltos no estaban ordenados por fecha, de manera que el resultado era erróneo ya que no devolvía los 3 últimos proyectos. Para solucionarlo decidí hacer una subconsulta con los datos que necesitaba, para posteriormente ordenar los resultados de forma correcta. Tras tener esta sentencia SQL realicé la correspondiente transformación a la sintaxis del *ORM Eloquent*. La realización de esta consulta y la transformación resultó ser una de las tareas más difíciles y largas realizadas en este bloque, debido a la complejidad de la consulta. A continuación, mostramos el resultado de la consulta, en el que se puede apreciar su dificultad:

```
$proyectos = \DB::table(\DB::raw("(SELECT `historicohoras`.`idproyecto`, `proyectos`.`nombre`,
    `historicohoras`.`updated_at` FROM `historicohoras` INNER JOIN `proyectos` ON
    `historicohoras`.`idproyecto` = `proyectos`.`id` WHERE `proyectos`.`eliminado` <> 1 AND
    (`proyectos`.`nombre` LIKE '%".$cadena."%') AND `historicohoras`.`idtrabajador` = ". Auth::user()->id."
    ORDER BY `historicohoras`.`updated_at` DESC LIMIT ".PHP_INT_MAX.") AS subQuery"))
    ->select('idproyecto', 'nombre')
    ->groupBy('idproyecto')
    ->orderBy('updated_at','DESC')
    ->limit(5)
    ->get(array('nombre'));
```

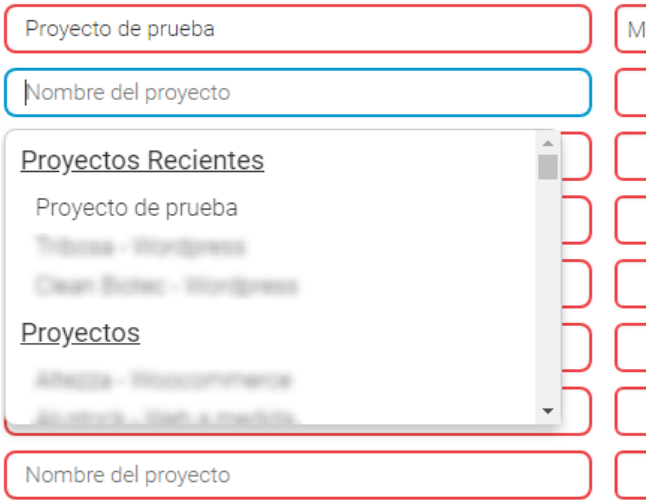
Figura 4.4 Consulta para la función `buscarPorNombreReciente`

- Por último, para que el plugin funcione correctamente en la aplicación, asigné, mediante *jQuery* (`$().bind()`), un evento al listado de proyectos, de manera que cada vez que se seleccione una de las sugerencias, se ejecute una función (ver Figura 4.5). Dicha función, se encargará de asignar el ID del proyecto seleccionado a un campo de tipo "hidden", para que el funcionamiento de la aplicación sea similar a como era en la versión anterior. Esto es debido a que las funciones *Ajax* de la aplicación requieren el ID de los proyectos para el correcto funcionamiento.

```
$('.typeahead').bind('typeahead:select', function(ev, sugerencia) {
    . . .
});
```

Figura 4.5 Asignación mediante JQuery

Para finalizar la parte del campo de sugerencias, debido a que se trata de un proyecto de reingeniería, y que la persona que desarrollo inicialmente la aplicación decidió que el funcionamiento del selector de fechas iba a realizarse mediante peticiones *Ajax*, tuve que actualizar el *JavaScript* correspondiente (`home.js`) para añadirle todo el código HTML escrito para esta tarea. De esta manera, cuando se refresque la página al cambiar el día se mantendrá el campo de sugerencias y no volverá a aparecer el `comboBox` antiguo. Tras esto, el campo de texto con las sugerencias se visualiza como en la siguiente imagen.



4.3.2 Creación del Pop-up

En la creación del botón que activará el pop-up que permita introducir una descripción a cada tarea, realicé una redistribución de la cuadricula de la tabla *Bootstrap*. En particular, le quité un pequeño trozo a las columnas dedicadas a las *Tareas*, *Módulos* y *Horas* de forma que pudiera incluir una nueva columna con el icono que desplegará el pop-up (ver Figura 4.6).

Figura 4.6 Redistribución de la tabla Bootstrap

Para introducir el icono de la etiqueta, utilicé *Bootstrap Glyphicon*, que nos permite usar una colección de iconos mediante clases CSS, como se puede ver en la siguiente figura:

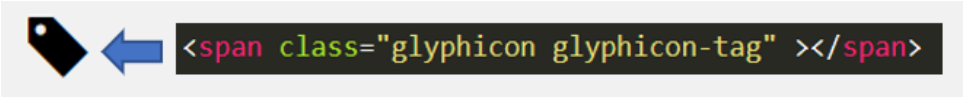


Figura 4.7 Uso de Bootstrap Glyphicon

Tras esto, realicé las modificaciones necesarias en la base de datos para que las tareas puedan guardar el campo de la descripción (básicamente añadir un nuevo campo a la tabla `historicoHoras`).

Seguidamente, para desarrollar el pop-up hice uso de *JQuery* para que, cuando se pulse en el icono, se le añada una clase mediante la función `.addClass()`, la cual cambie la visibilidad del elemento y así aparezca en pantalla. Para hacer desaparecer el pop-up una vez añadida la descripción, se ha hecho de forma similar mediante la función `.removeClass()`.

Tras esto apareció un problema, ya que actualmente la aplicación permite que se puedan insertar varias tareas a la vez. Esto me lleva a la necesidad de identificar cada pop-up para saber a qué tarea corresponde (ver Figura 4.8). En consecuencia, a la hora de asignar el evento al icono de la descripción habrá que pasarle un parámetro que indique qué pop-up se está abriendo.

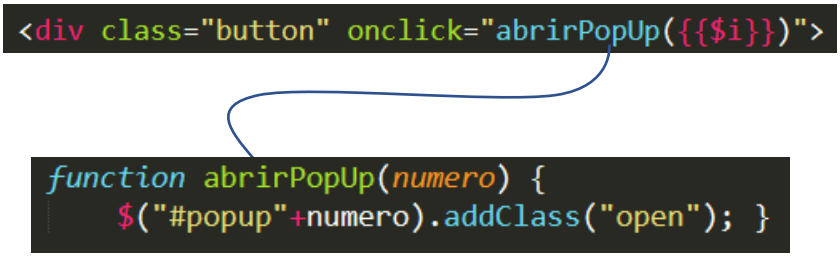


Figura 4.8 Asignación del evento `onClick` a la función `abrirPopUp`

Posteriormente, se llevó a cabo el desarrollo de la capa que se hará visible y donde se permitirá escribir la descripción, la cual consta de un título (`H2`) y un campo de texto donde irá la descripción. A dicha capa apliqué mejoras de CSS, como pueden ser:

- Uso de animaciones (mediante el selector *`animation`*). Por ejemplo, en el pop-up de la descripción aparecen unos círculos en la esquina superior izquierda y en la esquina inferior derecha, que tienen un pequeño movimiento de izquierda a derecha, para darle un poco de dinamismo a dicho pop-up.
- Uso de transformaciones (mediante el selector `transform`), el cual se ha usado para centrar la capa del pop-up en medio de la pantalla. Además también nos da la opción de reescalar elementos HTML, aunque en este proyecto no se ha utilizado.
- Uso de selectores CSS básicos para ajustar la forma y el color, de manera que el pop-up tenga un aspecto similar al resto de la aplicación.

En la siguiente Figura 4.9 se puede ver el resultado final del pop-up.

Figura 4.9 Versión final del pop-up

Finalmente, el cambio en la tabla `historicoHoras` requirió el ajuste de varias funciones de la aplicación que usaban dicha tabla. Además, se actualizaron los *JavaScript,* como hicimos anteriormente, para que tuvieran el HTML desarrollado en este apartado.

4.3.3 Mejoras de HTML/CSS

En este módulo el cliente ha solicitado varias mejoras. A continuación, se detalla cada una de ellas, así como la forma en la que se han abordado.

La primera de ellas consistía en mostrar en algún lugar de la pantalla el día en el que estamos introduciendo las tareas, con un formato especifico. Para implementar esta mejora utilicé la función *JavaScript* `toLocaleDateString()`, la cual admite dos parámetros:

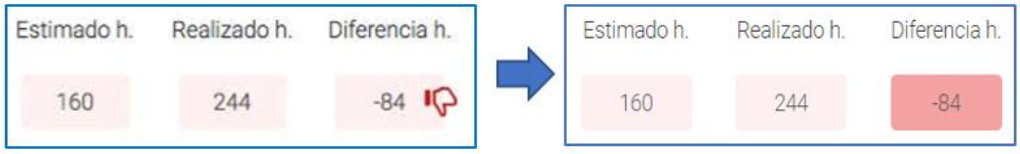
- El primero es un `Locale` (código que representa una región geográfica)
- El segundo es un array de opciones.

De esta manera, mediante las dos siguientes líneas de código, la mejora quedaría implementada.

```
var options = { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric' };
$('.fechaHome').text( now.toLocaleDateString("es-ES", options) );
```

La segunda de las mejoras a realizar consistía en eliminar el icono de la “mano” hacia arriba/abajo que aparece junto a la diferencia de horas entre las estimadas y las realizadas en los proyectos (ver recuadro verde de la Figura 4.6), para sustituirlo por algo más estético y minimalista.

Además, aprovechando esto, también se ha modificado el icono de la descripción asociada a cada tarea, de manera que, cuando el icono aparece en rojo, significa que la tarea incluye una descripción y, en caso de estar en negro, refiere a que la tarea no tiene una descripción. Para cumplir el requisito de los iconos de las “manos”, lo que he hecho es cambiar el color de fondo del campo de texto donde se muestran las diferencias de horas. De esta forma, la interfaz queda mucho menos recargada y más entendible a primera vista (ver la siguiente imagen con el resultado).



Versión antigua *Versión nueva*

La última de las mejoras realizadas en este módulo ha sido el cambio del estilo de los botones. Anteriormente, los botones apenas tenían un estilo CSS que siguiera la estética del resto de la aplicación. Ahora con la actualización he utilizado los selectores `:hover` y `:active`, de manera que los botones de la aplicación ahora pasan a tener el estilo que se puede ver en la siguiente figura.

Botón antiguo *Botón nuevo*

4.4 Pruebas

Tras acabar todos los requisitos que el cliente había planteado en esta fase, se ha comprobado la funcionalidad de la aplicación, insertando una tarea nueva con una descripción. Además, he comprobado que, al cambiar de fecha con el selector de días, la pantalla se refresca correctamente y el nombre del día parece adecuadamente, concluyendo que la actualización de todos los *JavaScript* se ha realizado correctamente. Todo esto se puede ver en las siguientes figuras:

- Comprobación de que la aplicación guarda tareas (el icono de descripción pasa a estar en rojo).

- Comprobación de que la tarea mantiene la descripción (se visualiza en el cuadro de texto de la descripción de la tarea en cuestión).

5. Fase 4. Mejoras en el módulo de proyectos.

En esta fase vamos a tratar todas las mejoras correspondientes al módulo de gestión de los proyectos.

5.1 Análisis

Para comenzar, comentaré que este módulo es utilizado tanto por los *trabajadores*, los cuales acceden con el usuario "user", como por los usuarios *administrador*, que acceden mediante "admin". Además, hay que recalcar que la vista del módulo será diferente en función del tipo de usuario que acceda al mismo, ya que un usuario *administrador* tiene más privilegios que un *trabajador*.

En particular, "user" utiliza este módulo para visualizar la lista de todos los activos que tiene la empresa, pero en ningún lugar puede consultar datos sensibles (presupuesto, precio por hora negociado…) ni modificar ninguna información sobre los proyectos. En el caso de "admin", este módulo le permitirá gestionar toda la información relacionada con los proyectos, ya que con él podrá tanto editar como eliminar y consultar las estadísticas de los proyectos.

Para este apartado, el cliente nos ha pedido es realizar unas mejoras en lo referente a la parte del administrador, de manera que se facilite la realización de sus tareas.

- La primera de ellas consiste en realizar una redistribución de la tabla donde se listan los proyectos, ya que el cliente nos explicó que actualmente los iconos/enlaces no están organizados de manera intuitiva y dan lugar a confusiones.
- La segunda, y más importante, consiste en hacer que la tabla donde se listan los proyectos pueda ser editable, esto quiere decir que cuando el administrador pulse un nuevo botón (a crearse), las propiedades del proyecto seleccionado (como por ejemplo el *nombre*, *categoría*, *presupuesto*, *fecha de fin*, etc...) pasen a ser campos de texto (*inputs*) donde su valor pueda ser cambiado de forma que los datos modificados sean también aplicados en la base de datos.
- Seguidamente, el cliente nos señaló que se debería hacer una actualización del filtro existente para buscar proyectos, ya que éste no funcionaba de una forma adecuada y además no disponía de los criterios de búsqueda necesarios.
- Por último, habrá que realizar mejoras en el código HTML y CSS para darle a este módulo un aspecto similar al del resto de la aplicación.

5.2 Diseño

Para llevar a cabo las mejoras solicitadas por el cliente, he realizado las tareas que se indican a continuación.

Respecto a la redistribución de la tabla, tras una reunión con el cliente llegamos al acuerdo de que la redistribución indicada en el siguiente boceto (ver Figura 5.1) se adecuaba a sus requisitos, evitando así las confusiones que produce la interfaz actual (ver Figura 5.2).

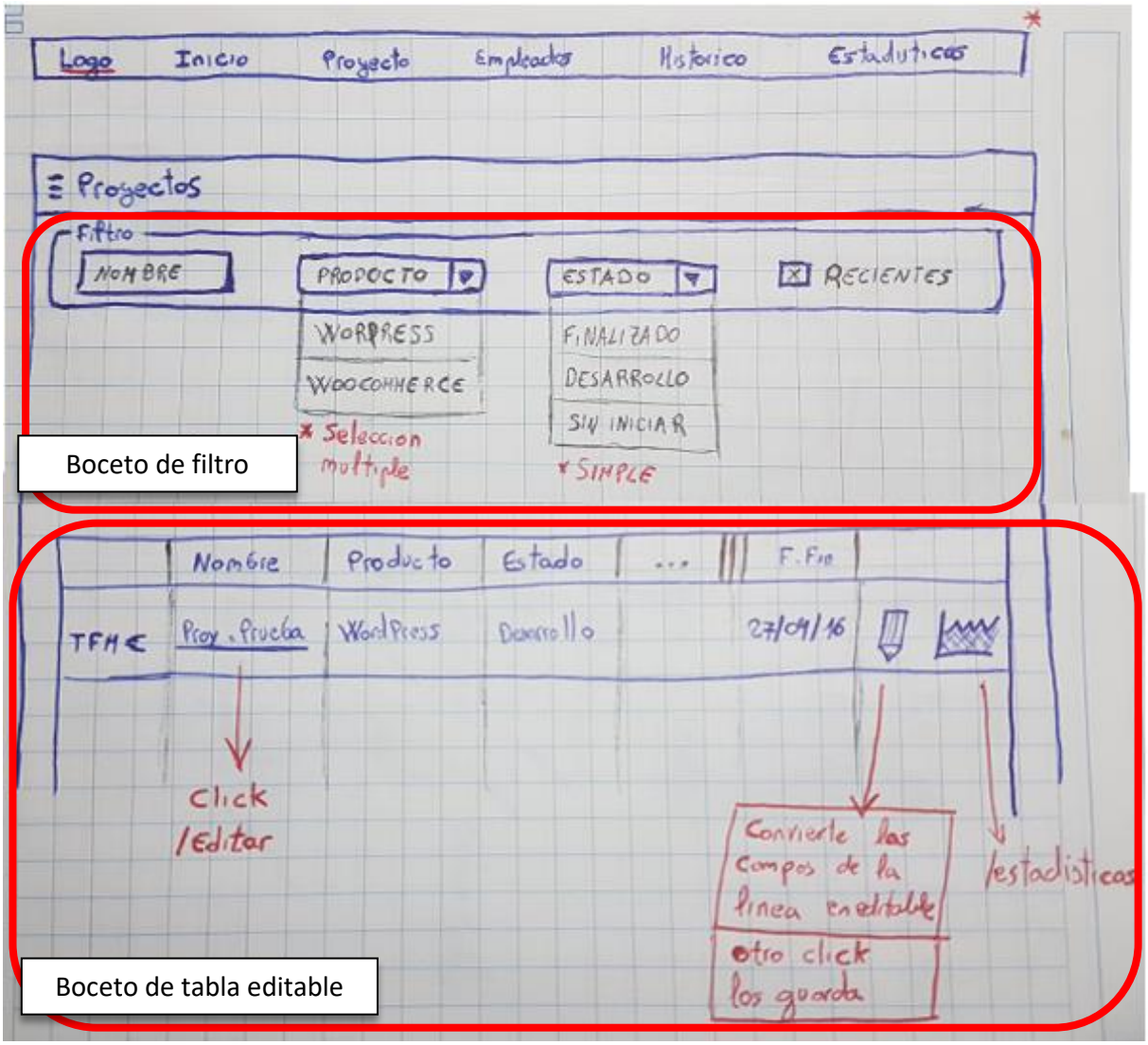


Figura 5.1 Redistribución de la tabla editable

Figura 5.2 Interfaz del módulo de proyectos (a mejorar)

Respecto a la posibilidad de convertir la tabla que lista los proyectos en editable, he encontrado un plugin de *JQuery* que se adecua perfectamente a las necesidades del cliente y que además permite introducir en la tabla a editar campos de tipo `text`, `select`, `number` o `date`.

En lo referente al filtro de búsqueda, desarrollaré uno basado en *jQuery*, haciendo uso de la función `hide()` para ocultar las filas de los proyectos que no coincidan con los criterios de búsqueda. Para ello seguiré el boceto diseñado en la parte superior de la Figura 5.1.

Finalmente, para la actualización del código HTML y CSS, crearé una cabecera mediante CSS para la tabla de los proyectos y además actualizaré los iconos de todo el módulo para darle un estilo más actual y acorde con el resto de la aplicación.

5.3 Implementación

A continuación, se describen los pasos realizados para implementar las tareas anteriores.

5.3.1 Redistribución de la interfaz

Para la redistribución de la interfaz, siguiendo el boceto anterior (ver Figura 5.1), he modificado la vista `proyectos.php`, de manera que el código HTML se genere dinámicamente en función del rol del usuario, mostrándose la tabla con contenido y funcionalidad distinta en base a dicho rol. Esto es debido a que un administrador tiene más privilegios que un trabajador ya que un trabajador no tendría que poder llegar a editar o ver datos sensibles de los proyectos. Esto lo he añadido como mejora porque en la versión anterior de esta aplicación, los trabajadores podían llegar a ver datos sensibles, ya que había un fallo de seguridad en las URL de la aplicación. Notaré que para controlar si el usuario se trata de un administrador he utilizado simplemente la línea:

```
@if (Auth::user()->rol == 'admin')
```

En lo que refiere a la siguiente mejora de usabilidad, el cliente estaba interesado en que las letras que hay en la primera columna de la tabla (ver primera columna de la Figura 5.2) le proporcionaran un resumen sobre cómo se encontraba el proyecto.
Antes, en la versión antigua, las letras de la primera columna contenían la información del proyecto y, en caso de querer ampliar esta información, había que ir al final de la tabla para ver una leyenda con su significado (como se puede ver en la Figura 5.3).

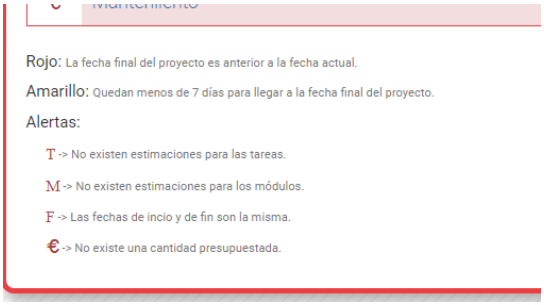


Figura 5.3 Leyenda con el significado de las letras

Como mejora pensé en crear una capa con todas las letras, de manera que, al situar el ratón sobre cualquiera de ellas, se pudiera leer la información (`title` de la capa) correspondiente a cada proyecto, tal y como se puede ver en la siguiente figura:

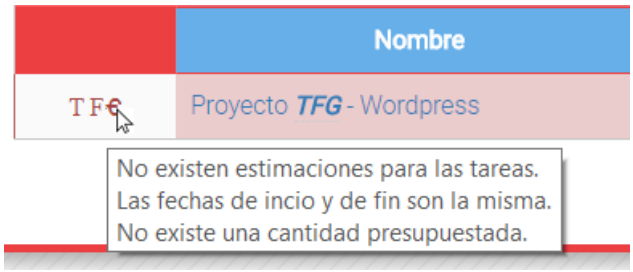


Figura 5.4 Concatenación de title

Tras la redistribución de la interfaz, la vista del administrador se quedó de la siguiente manera (teniendo en cuenta, que en este momento, el icono del lapicero no dispone de funcionalidad):

	Nombre	Categoría	Estado	Presupuestado	Horas estimadas	Horas realizadas	F. Fin	
MF€	Cliente A	A Medida	Desarrollo	0 €	48 h	23 h	09/02/15	
T	Cliente B	Web	Finalizado	101 €	80 h	83 h	07/11/16	
T€	Cliente C		Finalizado	0 €	151 h	558.25 h	10/07/15	
TF€	Cliente D	Web	Finalizado	0 €	1 h	10 h	08/05/17	

5.3.2 Creación de tabla editable

Una vez realizada la redistribución, he utilizado un plugin de *jQuery,* llamado *editable*, el cual transforma cada fila de una tabla, en una fila donde sus campos se convierten en editables una vez hecho click sobre el botón correspondiente (en este caso el botón del lapicero). En la siguiente figura se muestra dicho efecto.

Figura 5.5 Transformación de una fila a una fila editable

Para comenzar con la implementación, configuré el plugin con las necesidades del cliente. Para ello marqué las opciones del plugin de la siguiente manera:

```
236                 defaults = {
237                     keyboard: true,
238                     button: true,
239                     buttonSelector: ".edit",
240                     maintainWidth: true,
241                     dropdowns: {},
242                     edit: function() {},
243                     save: function() {},
244                     cancel: function() {}
245                 };
```

Figura 5.6 Configuración del plugin jQuery

Como se puede ver en la imagen anterior, indicamos que el botón inicial del plugin sea el elemento que contiene la clase "`edit`" (línea 239). Además, he indicado que el plugin tenga 3 funciones:

- `edit`, para que se pueda editar cada fila de la tabla.
- `save`, para guardar los cambios de la fila.
- `cancel`, que será para descartar los cambios realizados y que, debido a los requisitos del cliente, finalmente no utilizaremos.

Posteriormente, el plugin requiere configurar la función `init` (ver Figura 5.7), que es donde se definirá con qué teclas o botones se iniciará el plugin. Durante del desarrollo del proyecto surgió el siguiente problema. Como se puede ver en las líneas 264 y 269 de la Figura 5.7, la primera idea a la hora de configurar la función `init`, fue usar la función `bind()` para que el plugin se pudiese activar, y de esta forma, editar la fila tanto haciendo click sobre el icono de editar (línea 269) como haciendo doble click sobre la fila a editar (línea 264). No obstante, debido a que la vista `proyectos.php` es diferente para trabajadores y administradores, permitir editar al hacer doble click sobre la fila, constituía un fallo de seguridad, por lo que, en la versión final de la aplicación esta funcionalidad fue retirada, permitiendo solamente editar las filas mediante la selección del icono de editar.

```
258                 init: function() {
259                     this.editing = false;
260                     //Configuro para que se puede editar haciendo doble click sobre la fila
261                     if (this.options.dblclick) {
262                         $(this.element)
263                             .css('cursor', 'pointer')
264                             .bind('dblclick', this.toggle.bind(this));
265                     }
266                     //Configuro para que active mediante el boton definido
267                     if (this.options.button) {
268                         $(this.options.buttonSelector, this.element)
269                             .bind('click', this.toggle.bind(this));
270                     }
271                 },
```

Figura 5.7 Función `init`

Tras la configuración del plugin se dividirá en dos partes.

- La primera a desarrollar será la función `edit`, la cual se encargará de cambiar la fila de modo texto al modo editable. En su desarrollo se tendrá en cuenta cómo se cambian datos de tipo *text*, *comboBox* y *fechas*.
- La segunda funcionalidad que tendré que desarrollar viene dada por `save`, la cual devolverá la tabla a su estado normal (modo texto) y actualizará los cambios en la base de datos.

A continuación, explicaré cómo he desarrollado estas dos funciones.

En lo que se refiere a la función `edit`, hay que decir que el plugin se encarga de gestionar todos los inputs de tipo *text* (Figura 5.8), por lo que para estos campos de la tabla no habrá que escribir nada de código.

Figura 5.8 Input de tipo text

El problema surge cuando queremos configurar adecuadamente *comboBox* y los inputs que contienen una *fecha*. Como se muestra en la siguiente tabla (ver Tabla 5.9) el input de tipo select (*comboBox*) se construye a partir de `options.dropdows` , que es un array donde escribiremos las opciones que tendrá el *comboBox* (ver recuadro verde del modo editable en la Tabla 5.9).

Tras esto, para darle un aspecto similar a la información almacenada en la base de datos (antes, al ser un campo de texto, el estado de los proyectos estaba guardado de multiples formas, ya sea en mayúsculas, minúsculas o incluso sin espacios) ejecuté los siguientes UPDATES.

```
1. UPDATE `proyectos` SET `estado` = 'Desarrollo' WHERE
   `proyectos`.`estado` = 'desarrollo'
2. UPDATE `proyectos` SET `estado` = 'Finalizado' WHERE
   `proyectos`.`estado` = 'finalizado'
```

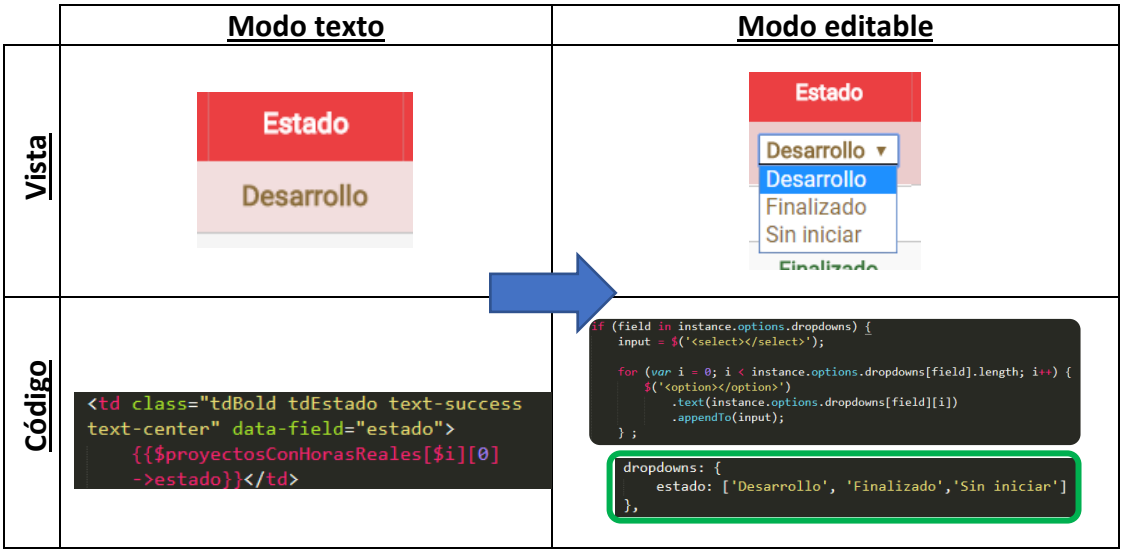


Tabla 5.9 Modo texto a ComboBox

Otro gran problema que he tenido al desarrollar la función `edit`, ha sido a la hora de tratar campos de tipo *fecha*. Como estoy interesado en darle a la aplicación un mismo estilo CSS (mismos colores, basados en una gama de rojos), he utilizado para el campo de tipo *fecha* el *JavaScript Pikaday* (`Pikaday.js`).

Para usar *Pikaday*, he tenido que importar su *JavaScript* y su CSS. Tras esto, el tipo *Pikaday* no era capaz de procesar el estilo de fecha que utiliza la aplicación, así que hice uso de las funciones `format` y `toString` (ver Figura 5.10) que nos proporciona este *JavaScript* para formatear y procesar dichas fechas.

```
pickers[this] = new Pikaday({
    field: $("td[data-field=fechaFin] input", this)[0],
    format: 'DD/MM/YY',
    toString(date, format) {
        // Formateo la fecha de la manera correcta, para que la
        aplicacion la entienda
        var dia = date.getDate()+"";
        var mes = "" + (date.getMonth() + 1);
        var anio = date.getFullYear()+"";
        if (mes.length < 2) mes = '0' + mes;
        if (dia.length < 2) dia = '0' + dia;
        var resultado = dia+"/"+mes+"/"+anio.substring(2, 4);
        return resultado;
    },
    parse(dateString, format) {
        // dateString es el resultado de `toString`
        const parts = dateString.split('/');
        const day = parseInt(parts[0], 10);
        const month = parseInt(parts[1] - 1, 10);
        const year = parseInt(parts[1], 10);
        return new Date(year, month, day);
    }
});
```

Figura 5.10 Función toString y parse de `Pikaday.js`

Tras la configuración adecuada de las funciones de `Pikaday.js`, el resultado final fue el siguiente.

Figura 5.11 Modo texto a modo editable en los campos de tipo *fecha*

Para el desarrollo de la función `save` he seguido el proceso reflejado en la Figura 5.12. En particular, en primer lugar, recogemos el valor de todos los campos (recoger el *value* del input) en la fila que estemos editando. Dichos datos se almacenan en un array. Mediante una petición *Ajax* (ver punto 1), la cual pasa por el fichero de rutas, se redirige a la función `guardarFila` (ver punto 2). Como se puede ver en el punto 3, en dicha función se recogen las variables enviadas mediante POST (ver líneas desde la 242 a la 247 del punto 3). Seguidamente, se actualizarán dichos datos mediante una instrucción de modificación (ver línea 253 del punto 3).

Todo este proceso queda reflejado en la siguiente Figura 5.12.

Figura 5.12 Diagrama para guardar los datos modificados de un proyecto

5.3.3 Creación del filtro de búsqueda

Para la implementación de este apartado, surgió un problema, para el cual el cliente me dio su aceptación. En primer lugar, diseñé e implementé un filtro siguiendo el boceto de la Figura 5.1. Posteriormente, el cliente cambió de opinión y me dijo que no estaba conforme con el filtro, ya que los criterios que tenía el filtro al final le parecían pocos. Por ello, tras una reunión y modificando algunos requisitos, volví a implementar otro filtro, que finalmente fué el que se está utilizando.

Por ello, este apartado lo dividiré en dos partes para mostrar cómo he realizado cada uno de los filtros, ambos basados en *jQuery,* pero completamente diferentes.

El primero de los filtros, que luego fue descartado, utilizaba las siguientes funciones de *jQuery*:

- `keyUp()` – Función para detectar cuándo se ha pulsado un tecla.
- `hide()`/`show()` – Función para ocultar/mostrar, respectivamente, las filas que coincidan en la búsqueda.
- Selector `contains` – Función con la que compararemos cadenas de texto.

En particular, para desarrollarlo, seguía la siguiente idea; cada vez que se detecta una pulsación (`keyUp`), se ocultan todas las filas de la tabla utilizando el siguiente código.

- `$("#my-table tbody>tr").hide();`

Posteriormente se muestran las coincidencias mediante la siguiente línea:

- `$("#my-table td:contains-ci('"+$(this).val()+"')").parent("tr").show();`

El segundo de los filtros utiliza un plugin de *JQuery* llamado `TableFilter` [2]. Este plugin se encarga de gestionar automáticamente todo lo referente al filtro (comparación de caracteres, listados de resultados, etc.). Lo único que hay que indicarle al plugin es la configuración que deseas y el tipo de dato (*texto*, *numero*, *fecha*) que hay en la columna donde deseemos implementar el filtro, ya que no es lo mismo un filtro para un texto que para una fecha.
A la hora de desarrollar este filtro tuve que tener en cuenta que la vista no es igual para los administradores que para los trabajadores, por lo que, en función del tipo de usuario, las opciones del filtro debían ser diferentes. Por ello, decidí que en función del tipo de usuario logueado, la tabla a visualizar tendría una clase CSS diferente: `admin`, para los administradores, o ninguna clase si el usuario logueado es `user`.
Así, mediante la comprobación de la clase (ver la primera línea de la Figura 5.13), puedo elegir las opciones que se necesitan cargar en el plugin.

```
if($("#my-table").hasClass( "admin" )){
    //Opciones para el filtro de administrador
    ...
}else {
    // //Opciones para el filtro de Usuario/Trabajador
    ...
}
//Inicio del filtro con las opciones correspondientes
tf.init();
```

Figura 5.13 Comprobación y carga de opciones en TableFilter

Tras esto, mostraré en la siguiente tabla cuáles son las principales opciones del plugin que he configurado para que el resultado final sea el deseado por el cliente.

```
//Opciones para el filtro de administrador
var tf = new TableFilter(document.querySelector('#my-table'), {
    ...
    col_0: 'select',
    col_2: 'select',
    col_3: 'select',
    col_4: 'select',
    col_8: 'none',
    col_9: 'none',
    col_10: 'none',
    col_11: 'none',
    rows_counter: {
        text: 'Proyectos encontrados: '
    },
    btn_reset: {
        text: 'Limpiar filtro'
    },
    auto_filter: {
        delay: 700
    },
    ...
    col_types: [
        'string', 'string', 'string',
        'string', 'number', 'number',
        'number', { type: 'date', locale: 'es',
                    format: ['{dd}-{months}-{yy}'] }
    ],
    custom_options: {
        cols:[4],
        texts: [[
            '0 - 150',
            '150 - 300',
            '300 - 450',
            '450 - 600',
            '600 - 750'
        ]],
        values: [[
            '>0 && <=150',
            '>150 && <=300',
            '>300 && <=450',
            '>450 && <=600',
            '>600 && <=750',
        ]],
        sorts: [false]
    },
    extensions:[{ name: 'sort' }]
```

Asignación del plugin a la tabla, mediante el ID.

Selección del tipo de filtro que se desea para la columna de la tabla.
En caso de `select`, las opciones podrán ser personalizadas.
En caso de ser `none`, esa columna no dispondrá de filtro.

Traducción de algunos textos del plugin al español.

Tiempo en milisegundos que tardará el filtro en buscar.
Con esta opción evitamos tener que pulsar la tecla "Intro" cada vez que queramos buscar.

Tipo de dato que irá en cada columna.

Como se puede observar, el tipo `date` se puede personalizar.

Valor personalizado para las columnas de la tabla que sean de tipo `select` (comboBox).

En la opción `text` se pondrán los valores a mostrar en la tabla.

En `values`, se pondrá la expresión a la que corresponde el `text` anterior.

Extensión del plugin que nos permite ordenar las columnas de la tabla sean ordenables.

Tabla 5.14 Principales opciones de configuración del plugin `TableFilter`

Tras configurar el plugin correctamente, la tabla de proyectos tendría el aspecto que se puede ver en la siguiente figura:

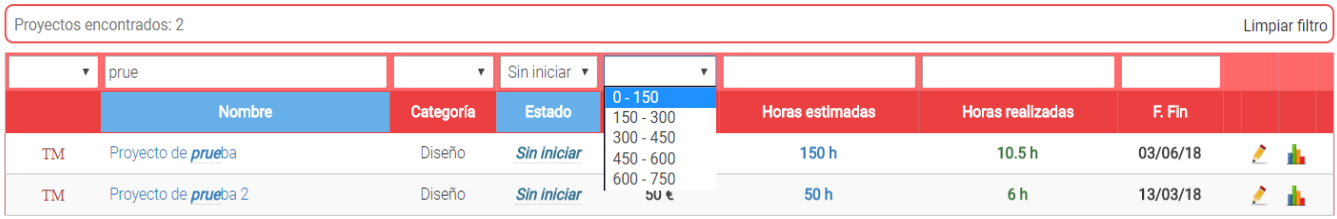


Figura 5.15 Aspecto final de la tabla de proyectos usando el plugin `TableFilter`

5.3.4 Mejoras de HTML/CSS

Para acabar con este módulo, he realizado unas últimas mejoras para darle un aspecto similar al de toda la aplicación.

La primera de ellas ha sido la modificación de la hoja de estilos `tableFilter.css` para que, tanto los filtros, como la capa superior de estos (recuadro donde se indica el número de proyectos encontrados), tengan una tonalidad en rojo como la del resto de la aplicación. Para ello ha sido necesario el uso de la declaración CSS `!important`, la cual corta la herencia de las hojas de estilos y da prioridad a la regla de estilo que tiene esta declaración. En la siguiente figura se puede ver una regla CSS de la aplicación donde ha sido necesario el uso de `!important`.

tablefilter.css

```
.fltrow td{border-bottom:1px solid #eb3f42 !important;
```

Lo siguiente ha sido eliminar el código HTML correspondiente al filtro de la aplicación antiguo (el que ya existía en la aplicación) y sustituirlo por una cabecera hecha a base de código CSS, para evitar una sensación de vacío y ayudar a una rápida identificación del módulo en el que nos encontramos. Para conseguir este efecto de la cabecera, las principales reglas de estilo utilizadas han sido:

- Para transformar el texto a mayúsculas;
  `text-transform: uppercase;`
- Para que el navegador priorice la legibilidad sobre la velocidad de renderizado.
  `text-rendering: optimizeLegibility;`
- Para crear sombras detrás del texto
  `text-shadow: 1px -2px 0px #3F3E3E, -1px 2px 1px #788F9E, -2px 4px 1px #788F9E;`

Por último, he actualizado los iconos del módulo por otros más modernos y fáciles de comprender a primera vista.
Tras finalizar estas tareas, el módulo de proyectos se vería de la siguiente manera:

Figura 5.11 Versión final del módulo de proyectos

5.4 Pruebas

Tras realizar las pruebas correspondientes, vemos que tanto los enlaces de la redistribución, como el plugin del filtro, funcionan correctamente. Igualmente, funciona la tabla editable con el listado de los proyectos.

6. Mejoras individuales

En este punto del TFG trataré las mejoras que se encuentran tanto dentro como fuera de los módulos tratados anteriormente. El único punto que diferencia estas mejoras con respecto a las anteriores es que no son necesarias para el funcionamiento de la aplicación, sino que son peticiones del cliente para mejorar la usabilidad.

6.1. Edición de la barra de navegación

Como se puede ver en la Figura 6.1, la barra de navegación que existía en la aplicación anterior era genérica para cualquier tipo de empresa. Por ese motivo, he decidido modificar tanto su HTML como su CSS, incluyendo:

- Una barra con un estilo de letra más definido (antes las letras se veían difuminadas).
- Unos iconos más acordes a la aplicación.
- El logo de la empresa para hacer aún más personal el resultado final.
- Menús desplegables con algunas opciones interesantes, como puede ser crear proyectos.

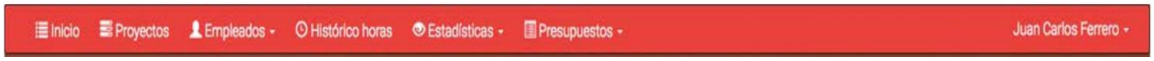


Figura 6.1 Barra de navegación antigua

En la siguiente Figura 6.2 se puede ver cuál ha sido el resultado final de la barra de navegación de la aplicación.

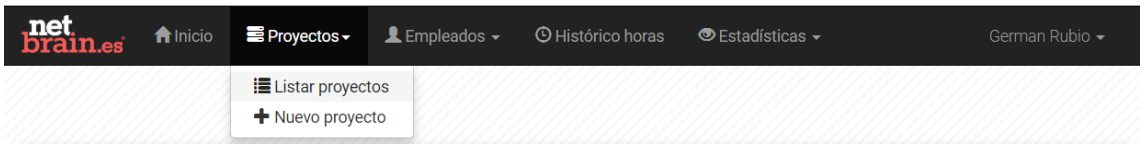


Figura 6.2 Barra de navegación de la aplicación

6.2.Realización de una página de error para la aplicación

Debido a que la aplicación anterior no dispone de una página de error que nos informe del estado en que se encuentra la aplicación y nos permita volver a un estado seguro, he creado una vista que corresponderá a la página de error de la aplicación.

Con dicho objetivo en mente, he creado una capa central (`<div>`) en donde se mostrará el mensaje del error, y en el cual haya un botón con el que se pueda volver hacia atrás.

Una vez desarrollado el código de la capa, le he dado un retoque a la vista mediante CSS para darle un estilo similar al resto de la aplicación.

Tras crear la página de error, se han actualizado todas las funcionalidades de la aplicación para que, en caso de error, se redirijan a esta vista. Además, se han controlado los errores de las peticiones GET con objeto de evitar excepciones en la aplicación (por ejemplo, en caso de que un usuario intente editar un proyecto/trabajador que no exista o no tenga permisos, mediante una URL, ahora la aplicación redigirá a la nueva vista de error).

A modo de ejemplo, en la Figura 6.3, se describe el funcionamiento de la página de error. En primer lugar decir que el diagrama está realizado poniendo como ejemplo la edición de un proyecto cuya ID no existe, por lo que, como se puede ver en la línea 3 del paso 1, la variable `proyecto` es nula. Seguidamente, el controlador de los proyectos hará la comprobación para ver si el proyecto no existe o esta "eliminado" (ver línea 5 del paso 1) y nos redigirá a la página de error (ver línea 6 del paso 1), a la que le llegará la variable `message` enviada desde el controlador, la cual será mostrada dentro de una etiqueta `H2` en la vista (como se puede observar en la línea 2 del paso 2). De esta forma, el resultado final mostrado por el navegador será indicado el punto 3.

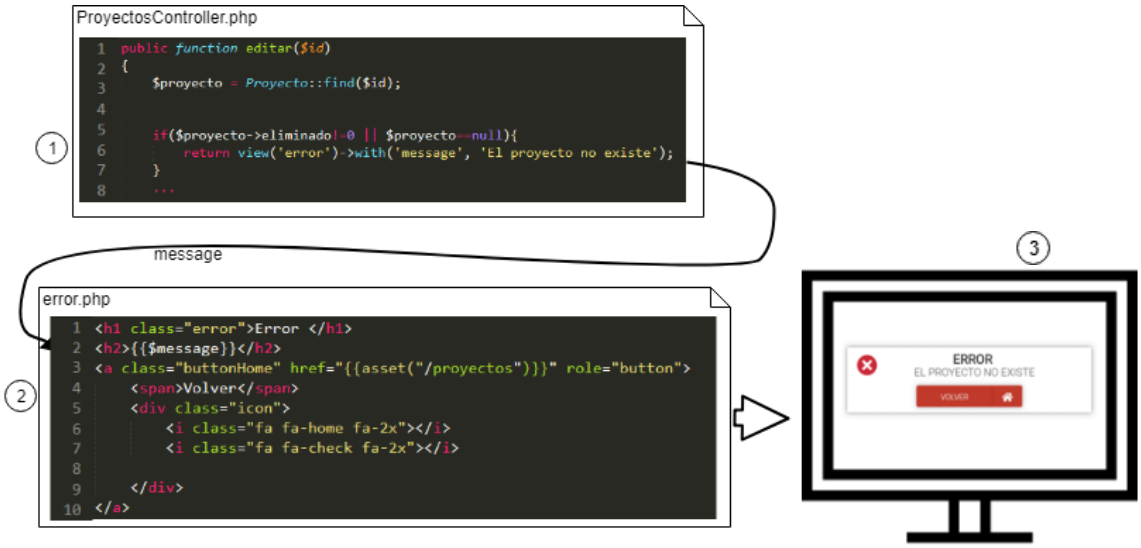


Figura 6.3 Diagrama de funcionamiento de la página de error.

6.3.Creación de fichero de configuración

Debido al interés que tenía el cliente porque se le facilitase el mantenimiento de la aplicación, se ha decidido utilizar ficheros de configuración. De esta manera, la aplicación obtendrá de estos ficheros los datos para rellenar algunos `comboBox` (los que el cliente considera que cambiarán con el tiempo). Así, en caso de que en un futuro se quieran añadir nuevas opciones, sólo habrá que añadirlas a dicho fichero.

Ya que *Laravel* dispone de gestión de ficheros de configuración, voy a hacer uso de ellos.

A modo de ejemplo, en la Figura 6.4 se explica el proceso de uso de estos ficheros de configuración. En primer lugar, habrá que rellenar el fichero con los valores que deseemos, siguiendo el formato que se puede ver en el paso 1. Para utilizar estos ficheros dentro de un controlador habrá que hacer uso de la línea de código que se puede ver en el paso 2, la cual nos devuelve un array con los valores escritos en el fichero. Por último, habrá que enviar dicho array a la vista correspondiente, donde se recorrerá para construir los inputs (`comboBox`) como se puede ver en el paso 3. Tras finalizar este proceso el resultado será el que se puede ver en el paso 4 de la Figura 6.4.

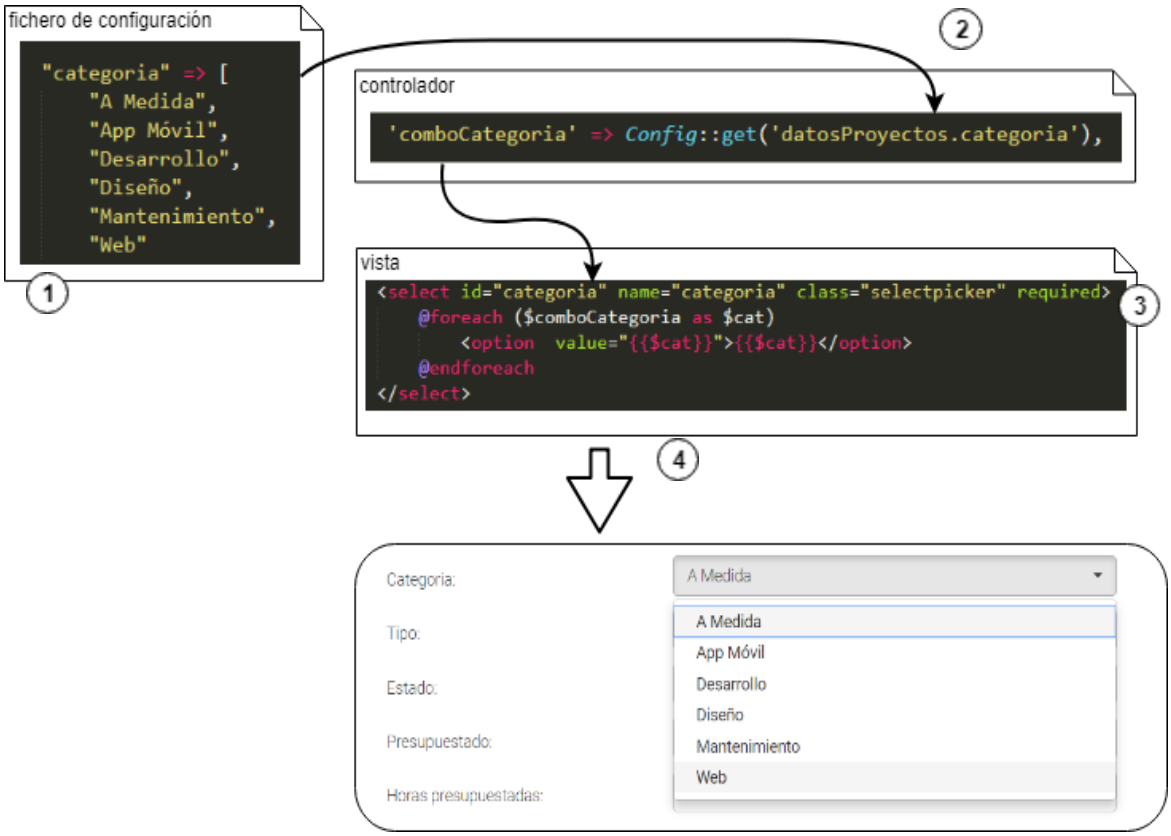


Figura 6.4 Uso de fichero de configuración

6.4. Mejoras con *JQuery*

En este apartado hablaré de otras dos mejoras que he realizado en la aplicación, para las cuales he usado, de nuevo, *jQuery*.

- La primera de ellas consiste en facilitar, cuando se ha hecho scroll hasta un cierto punto dentro de la ventana del módulo de proyectos, que el usuario pueda subir fácilmente hasta arriba de la página pulsando un botón. Para ello, he pensado en crear un botón flotante (ver Figura 6.5) en la tabla de proyectos el cual se hace visible a partir de hacer scroll cierto número de pixeles hacia abajo. Al pulsar este botón, mediante la función `window.scrollBy()`, se vuelve al principio de la tabla sin necesidad de tener que hacer scroll hacia arriba.

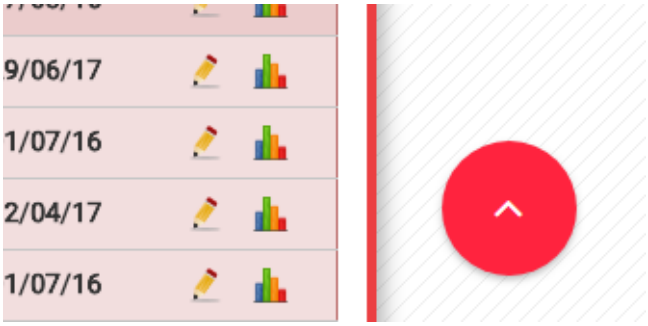
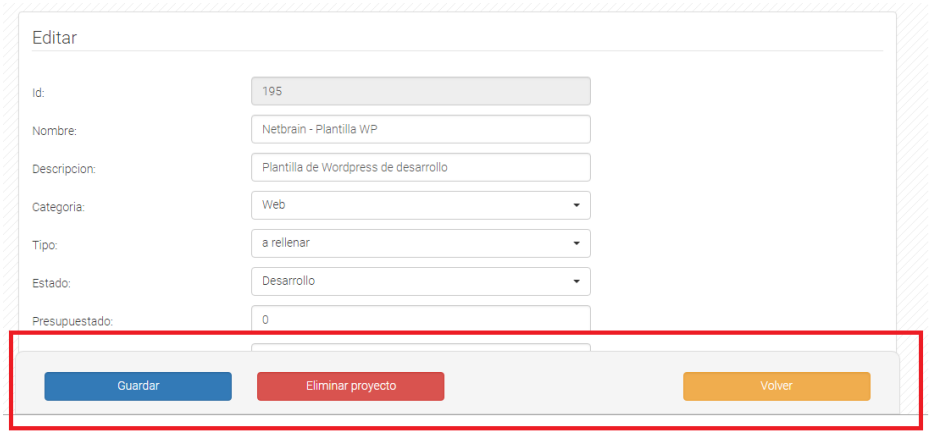


Figura 6.5 Botón para evitar hacer scroll

- La segunda mejora se da también dentro del módulo de proyectos y, en particular, dentro del apartado de editar proyectos. Su realización vino motivada porque muy frecuentemente, el administrador edita algún dato de los proyectos, no obstante, debe bajar hasta el final de la página para darle a los botones de *Guardar*/*Eliminar proyecto*/*Volver*. Mediante esta mejora, evitaremos que el usuario haga scroll hasta el final de la página para darle al botón de *Guardar*/*Eliminar proyecto*/*Volver*. Esta mejora consiste en crear una capa flotante que contenga los botones de *Guardar*/*Eliminar proyecto*/*Volver* (ver parte superior de la Tabla 6.6), para evitar tener que hacer scroll cuando solamente queramos cambiar un dato. Esta capa flotante debe desaparecer si llegamos a hacer scroll hasta el final de la página (ver parte inferior de la Tabla 6.6). Para implementar esta mejora, he utilizado el evento `scroll`, de manera que compruebo la posición donde se encuentra la visualización la página. Dependiendo de esta posición, ejecuto la función de *jQuery* `slideUp()` o `slideDown()` que permiten mostrar/ocultar, respectivamente la capa con una animación.

Antes de Scroll


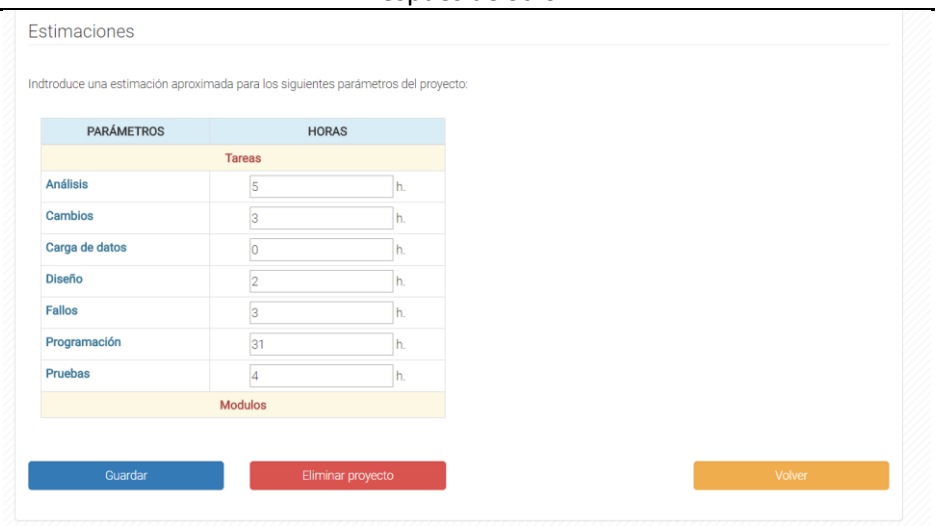
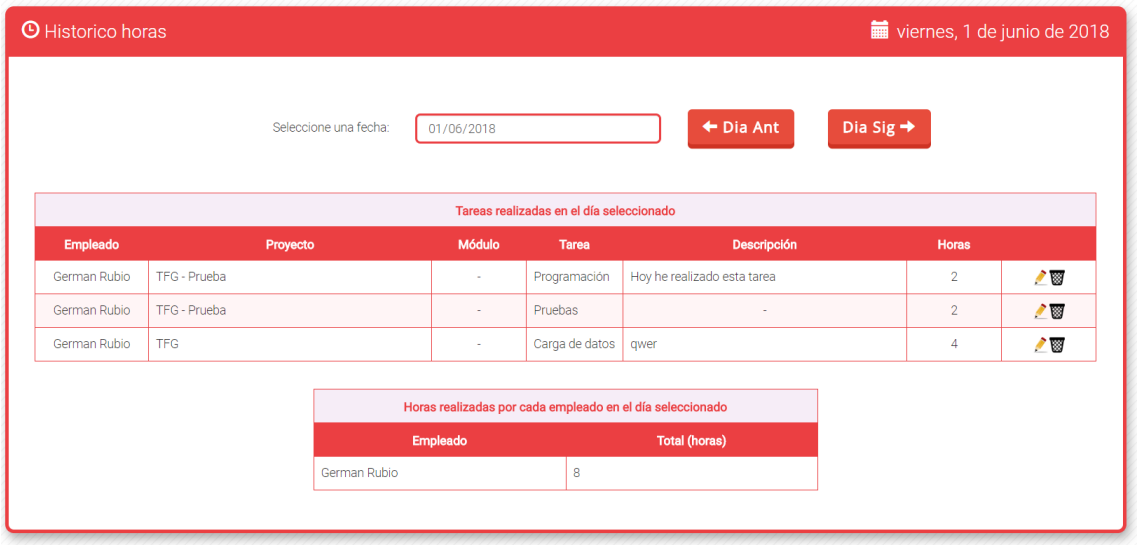
Después de Scroll


Tabla 6.6 Capa flotante antes/ después de hacer scroll

6.5.Actualización del módulo de gestión de horas

Debido a que ahora las tareas también pueden tener una descripción, hay que actualizar todo el código HTML de este módulo para que las descripciones sean listadas y el administrador pueda editarlas. En la siguiente figura se puede ver la versión final de este módulo.



6.6.Actualización del estilo de la aplicación

Tras finalizar todo lo que el cliente requería, actualicé toda la aplicación para darle un estilo similar. Entre estas actualizaciones están las siguientes:

- Estilo de los botones.
- Estilo de las tablas.
- Poner el filtro `TableFilter` a otras tablas de la aplicación.
- Cambio del fondo de pantalla de la aplicación.

No merece la pena destacar ningún aspecto sobre el desarrollo de estas dos secciones 6.5 y 6.6, ya que, en este punto del TFG, el estudiante ya estaba familiarizado con las tecnologías usadas y su desarrollo no le supuso un gran reto.

7. Comparativa entre el tiempo estimado y el real

En la siguiente tabla podemos ver el identificador de cada tarea, el tiempo estimado para su realización y el tiempo real invertido.

Id.	Nombre	Horas estimadas	Horas reales
1.1	**Planificación del proyecto**	**81 horas**	**101 horas**
1.1.1.	Análisis de requisitos	5 horas	5 horas
1.1.2.	Planificación	16 horas	16 horas
1.1.3.	Memoria TFG	60 horas	80 horas
1.2.	**Estudio de tecnologías**	**24 horas**	**24 horas**
1.2.1.	Análisis	18 horas	18 horas
1.2.2.	Prueba	6 horas	6 horas
1.3.	**Corrección de errores**	**25 horas**	**25 horas**
1.3.1.	Error en módulo de estadísticas	5 horas	5 horas
1.3.2.	Error ordenar por fechas	5 horas	5 horas
1.3.3.	Error al eliminar proyectos	15 horas	15 horas
1.3.3.1.	Análisis	1 hora	1 hora
1.3.3.2.	Diseño	2 horas	2 horas
1.3.3.3.	Implementación	10 horas	11 horas
1.3.3.4.	Pruebas	2 horas	1 hora
1.4.	**Mejoras de usabilidad y rediseño de la interfaz**	**165 horas**	**167 horas**
1.4.1	Mejoras página principal	47 horas	51 horas
1.4.1.1.	Análisis	5 horas	5 horas
1.4.1.2.	Diseño	5 horas	5 horas
1.4.1.3.	Implementación	35 horas	40 horas
1.4.1.4.	Pruebas	2 horas	1 horas
1.4.2	Mejoras en el módulo de proyectos	68 horas	77 horas
1.4.2.1.	Análisis	5 horas	5 horas
1.4.2.2.	Diseño	5 horas	5 horas
1.4.2.3.	Implementación	55 horas	65 horas
1.4.2.4.	Pruebas	3 horas	2 horas
1.4.3	Mejoras individuales	50 horas	39 horas
1.4.3.1.	Análisis	5 horas	3 horas
1.4.3.2.	Diseño	5 horas	3 horas
1.4.3.3.	Implementación	35 horas	30 horas
1.4.3.4.	Pruebas	5 horas	3 horas
1.5.	**Realización de un manual**	**5 horas**	**5 horas**
	Horas totales	**300 horas**	**322 horas**

Como se puede ver en el tiempo real invertido, apenas ha habido una desviación considerable. No obstante, sí que hay que destacar dos grandes desviaciones del proyecto:

- La primera está relacionada con la consulta realizada en el punto 1.4.1.3. (página 22) y la conversión de esta a la sintaxis del *ORM Eloquent*. El motivo fue debido a la complejidad de la misma.
- La segunda, y más importante, se dio por el cambio de opinión por parte del cliente (explicado en el punto 1.4.2.3., página 34), lo que supuso una desviación de 10 horas. Dicho tiempo pudo ser recuperado tras reunirme con el cliente para valorar los requisitos más importantes del punto 1.4.3.

Además de estas dos desviaciones, también hay que comentar que ha habido una desviación en la realización de la memoria, que ha sido debida, a los problemas con la redacción (relacionados con la poca experiencia que tengo redactando grandes textos).

En resumen, el proyecto ha durado 22 horas más de lo previsto, de las cuales la mayoría han sido por la redacción de la memoria. Aun así, considero que es una desviación más que aceptable.

8. Conclusiones

En este apartado se realizará una reflexión sobre todos los aspectos relacionados con el desarrollo de este Trabajo Fin de Grado.

- Como punto principal, nombrar que este proyecto me ha acercado al mundo laboral, ya que ha sido desarrollado en un entorno muy similar al que me espera en el futuro. Esto ha supuesto un gran reto para mí, ya que se trata de un proyecto de reingeniería, lo que quiere decir que no solo hay que pensar una solución para los nuevos problemas, sino que también requiere comprender el código desarrollado por otros programadores. Gracias a ello, me he dado cuenta de la importancia de comentar y documentar el código desarrollado, tanto para bien de uno mismo, como para su uso y compresión por parte de otros desarrolladores que deban usarlo. Además, debo decir que este TFG me ha permitido darme cuenta de lo que es trabajar con un cliente real el cual te vaya proponiendo nuevos requisitos una vez comenzado el proyecto, los cuales debes valorar si pueden realizar.
- Por otra parte, también he aprendido a desarrollar aplicaciones web con *PHP*, un lenguaje que no se ve durante la carrera. Además, he aprendido a utilizar el framework de *PHP Laravel*. También he utilizado otras tecnologías actuales como *JavaScript, jQuery, Ajax* y *CSS*.
- Otro de los aspectos de los que me he visto beneficiado tras la realización de este TFG ha sido a la hora de buscar información sobre problemas que iban surgiendo durante el proyecto, ya que cada vez solucionaba los problemas más rápido debido a que iba aprendiendo a refinar las búsquedas por internet.
- También he aprendido a planificar mejor las reuniones, para así sacar un mayor provecho en un menor tiempo.
- Como punto final, quiero destacar que estamos muy satisfechos tanto el cliente como yo con el trabajo realizado. Además, por mi parte, también estoy muy agradecido con los conceptos y lecciones aprendidas tras finalizar el TFG.

9. Bibliografía

Notaremos que la bibliografía que se muestra a continuación ha sido consultada a lo largo de todo el desarrollo del proyecto, durante los meses de febrero a mayo de 2018.

- PHP
  http://php.net/manual/es/
- Laravel
  https://laravel.com/docs/5.0
  https://styde.net/laravel-5/
- JQuery
  http://api.jquery.com/
- AJAX
  https://www.w3schools.com/
  http://api.jquery.com/jquery.ajax/
- HTML & CSS
  https://www.w3schools.com/
  https://codepen.io/
- Plugins y configuración
  [1] https://github.com/twitter/typeahead.js
  [2] https://github.com/koalyptus/TableFilter
- Dudas Generales
  https://stackoverflow.com/